
THE UNIVERSITY *of* EDINBURGH

# Argumentation-Based Methods for Multi-Perspective Cooperative Planning

*Alexandros-Sotiris Belesiotis*


Doctor of Philosophy

Centre for Intelligent Systems and their Applications
School of Informatics
University of Edinburgh

2012

# Abstract

Through cooperation, agents can transcend their individual capabilities and achieve goals that would be unattainable otherwise. Existing multiagent planning work considers each agent's action capabilities, but does not account for distributed knowledge and the incompatible views agents may have of the planning domain. These divergent views can be a result of faulty sensors, local and incomplete knowledge, and outdated information, or simply because each agent has conducted different inferences and their beliefs are not aligned.

This thesis is concerned with Multi-Perspective Cooperative Planning (MPCP), the problem of synthesising a plan for multiple agents which share a goal but hold different views about the state of the environment and the specification of the actions they can perform to affect it. Reaching agreement on a mutually acceptable plan is important, since cautious autonomous agents will not subscribe to plans that they individually believe to be inappropriate or even potentially hazardous.

We specify the MPCP problem by adapting standard set-theoretic planning notation. Based on argumentation theory we define a new notion of plan acceptability, and introduce a novel formalism that combines defeasible logic programming and situation calculus that enables the succinct axiomatisation of contradictory planning theories and allows deductive argumentation-based inference.

Our work bridges research in argumentation, reasoning about action and classical planning. We present practical methods for reasoning and planning with MPCP problems that exploit the inherent structure of planning domains and efficient planning heuristics. Finally, in order to allow distribution of tasks, we introduce a family of argumentation-based dialogue protocols that enable the agents to reach agreement on plans in a decentralised manner.

Based on the concrete foundation of deductive argumentation we analytically investigate important properties of our methods illustrating the correctness of the proposed planning mechanisms. We also empirically evaluate the efficiency of our algorithms in benchmark planning domains. Our results illustrate that our methods can synthesise acceptable plans within reasonable time in large-scale domains, while maintaining a level of expressiveness comparable to that of modern automated planning.

# Acknowledgements

First of all, I offer my gratitude to my primary supervisor, Michael Rovatsos, for his excellent guidance, support and patience throughout this process. I thank my secondary supervisor, Iyad Rahwan, for his ideas, advice and hospitality during my visit to BUiD. I would also like to thank my examiners Simon Parsons and David Robertson whose comments and suggestions have improved the quality of this thesis. I thank the researchers in CISA whose advice and criticism provided direction to my work, and in particular Ron Petrick for his feedback. Many thanks to my friends and fellow members of the Agents Group: George Christelis, Matt Crosby, Francesco Figari, Tommy French, Areti Manataki and Iain Wallace. Finally, I am grateful to my family for their support and encouragement, and my parents, for everything.

# Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Alexandros-Sotiris Belesiotis*)

# Table of Contents

# Chapter 1

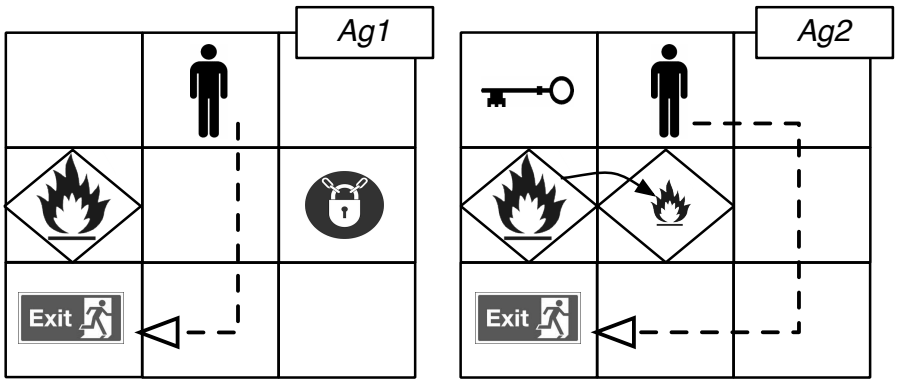
# Introduction

Modern computer systems are becoming increasingly complex and interconnected, providing an extensive infrastructure used by multiple human users and artificial entities. Interesting problems arise as a result of the activities and interactions among such entities. *Multiagent systems* research (Wooldridge, 2002; Weiss, 1999) is concerned with the individual reasoning tasks and the interactions among autonomous entities called agents. *Agents* perform actions in order to bring about their goals, and since in many cases individual actions are not sufficient, they form plans involving sequences of actions.

In complex domains, achieving one's own goals may not be always feasible. This can be the result of negative interactions with other agents, lack of information, limited individual capabilities, or requirements for joint effort (e.g. lifting a heavy object). To overcome these issues, agents can coordinate to minimise the negative interactions among their activities or maximise synergies. In a similar fashion, cooperative agents can collaborate by following joint plans in order to achieve common goals. Cooperation and coordination enable individual agents to transcend their individual capabilities and achieve goals that would be unattainable otherwise.

Multiagent planning algorithms (de Weerdt and Clement, 2009) address problems such as synthesising a plan for multiple, independently and often concurrently acting agents, or coordinating multiple, independently computed individual plans. Existing multiagent planning work usually takes account of distributed action capabilities, but not of distributed knowledge and the incompatible views agents may have of the planning domain. And yet, there are many situations where this problem can arise in complex multiagent scenarios. For example, this can be the result of faulty sensors, local and incomplete information, outdated beliefs, or simply because different agents have

Figure 1.1: Motivational example of a MPCP problem. Two agents *Ag1* and *Ag2* must come up with a plan that navigates the robot safely to the exit. The left figure depicts the planning knowledge of *Ag1*, whereas the picture on the right illustrates the beliefs of *Ag2*




conducted different inferences and their beliefs are not aligned. Cautious autonomous agents will not subscribe to plans that they consider to be harmful, ineffective or questionable. In order to reach agreement, agents need to align their knowledge and plans using mutually acceptable information.

Argumentation (Dung, 1995) has attracted much attention as a technique for resolving conflicts between agents, mainly due to its strong logical foundation and its suitability for use in multiagent situations. In previous work, argumentation methods have been proposed for problems related to multi-agent coordination, deliberation and practical reasoning (Sycara, 1989; Kraus et al., 1998; Parsons et al., 1998; Atkinson and Bench-Capon, 2007b; Amgoud et al., 2011).

This thesis proposes an argumentation-based approach that enables agents to resolve conflicts in their planning beliefs, and reach agreement on plans. We call this process *Multi-Perspective Cooperative Planning under ontological agreement* (MPCP). It deals with the problem of synthesising plans for multiple agents which have different, potentially conflicting views of the planning domain.

## 1.1 Motivating Example

Consider the example depicted in Figure 1.1. Two agents *Ag1* and *Ag2* share the goal of leading a robot safely to the exit location in a grid-world. They share informa-

tion regarding the location of the robot and the exit. Also, they both believe that the middle-left location is on fire. However, *Ag2* holds a more accurate specification of the behaviour of the fire, and believes that it is not safe to cross a location containing flammable objects (depicted by the square), if the nearby location is on fire. Also, *Ag2* holds the belief that there is a locked door in middle-right grid location, which is something *Ag1* is unaware of.

Based on their individual beliefs, the agents come up with different plans. *Ag1* synthesises the plan $\langle$*down, down, left, exit*$\rangle$, whereas *Ag2* comes up with the plan $\langle$*right, down, down, left, left, exit*$\rangle$. Unless one agent can persuade the other about their beliefs (e.g. *Ag2* persuading *Ag1* that the door in the middle-right location is not locked), both agents will not accept the plan synthesised by the other party. If they utilise their collective beliefs, they may reach the decision to follow the alternative plan $\langle$*left, pickup, right, right, down, unlock, down, left, left, exit*$\rangle$.

## 1.2 Multi-Perspective Cooperative Planning

Take the standard planning problem specification $P = \langle F, I, O, G \rangle$, with fluents $F$ that are used to describe an environment state as a conjunction of logical literals, an initial state $I$, operators $O$ that describe the agents' actions in terms of how they transform states, and a goal state $G$. This thesis investigates the problem of synthesising plans that can be defended against all possible objections based on a collection of individual specifications $P_i$ of the same planning problem, assuming that the fluent sets $F_i$ and $G_i$ are shared. In other words, we address the problem of *multi-perspective cooperative planning under ontological agreement*.

> *Assume a coalition of planning agents, each with a private set of beliefs describing the initial state of the domain and the specification of the planning operators, and assume that they are trying to achieve a shared goal. Construct a plan (i.e. a sequence of actions), that is entailed from the collective beliefs of the agents, and that can be defended against all objections.*

The MPCP problem arises in multiagent systems due to inconsistent views agents have about the planning domain. MPCP is also relevant from an individual agent perspective when multiple, potentially contradicting, sources of information are available. Even though MPCP is a multiagent planning problem, as it involves multiple agents seeking agreement on plans, the plan discussed might be a single-agent plan. Depending on the

context, solutions may be synthesised in a centralised or distributed setting, and these solutions may involve actions that are executed by multiple agents. The employed model involves a classical, sequential plan representation.

MPCP supports distributed action execution, provided that the resulting plan is encoded as a sequence of ground planning operators. Distributed action execution can be encoded by reserving a term, within every planning operator, to encode the agent that will perform this action. Equivalently, joint actions (e.g. lifting a couch) can be represented using a vector of agents. The language does not involve concurrent actions and does not address synchronisation issues, but everything that can be represented as a sequential plan is possible. A restricted form of concurrency can be supported within the limits of existing, conventional planning domain transformations (e.g. simple-time transformations). This model allows the representation of the action capabilities of heterogenous agents in the form of action preconditions (e.g. agents belong to classes, and an action is only possible to be executed by agents of a specific class). Similar to other aspects of the MPCP problem, the agents may hold contradictory views about such information.

The formal specification of the problem depends on the following:

**Formalism:** The formalism specifies the representation of the MPCP problem. It is related to the language that is used to specify planning domains and the structure of the axiomatised theories. An important characteristic of the formalism is its expressiveness, which governs its ability to encode rich domains in a succinct representation, and imposes additional requirements on the reasoning mechanism.

**Plan derivation:** Given a collective planning theory, which summarises the agents' beliefs about the state of the world and the specification of the planning, a derivation mechanism must specify the conditions under which plans are entailed from the agents' planning theories, even if objections to these plans exist.

**Plan acceptability:** Due to the potentially contradictory nature of the agents' beliefs, plan derivation cannot in itself define which plans should be accepted. The notion of plan acceptability specifies how derived plans are evaluated against potential objections that arise from the agents' theories.

**Knowledge:** Planning problems usually take into account structured information about the initial state and the operator specifications. The resolution of con-

tradictions may require additional beliefs regarding the credibility and sources of these pieces of information. Such meta-information does not appear in the agents' planning problem specifications.

### 1.2.1 Argumentation-Based Methods for MPCP

MPCP can be considered a compound problem consisting of a planning and a decision-making sub-problem. The planning problem involves the synthesis of plans that are entailed by the information held by the agents. This information is potentially contradictory, i.e. may lead to contradicting conclusions, which leads to the second sub-problem. The agents must evaluate whether plans can be defended against the objections that can be raised from the agents' beliefs. This way, we view MPCP as a planning problem with the additional constraint of plan *acceptability*.

In order to solve the first problem, we adapt the classical planning problem by relaxing the implicit assumption that the planning beliefs are non-contradictory, i.e. that they do not entail contradicting conclusions, and allow the agents to hold different specifications of the same planning operators. The resulting set-theoretic planning formalism enables planning with states which may contain contradicting information. Solutions to this problem are *candidate plans*, i.e. potential solutions that can be derived from the planning beliefs.

The decision-making sub-problem of MPCP is based on the specification of plan acceptability, which defines when it is rational to accept a candidate plan. In order to formalise this notion based on deductive argumentation, we propose *defeasible situation calculus*, a novel formalism based on the combination of situation calculus and defeasible logic programming, which allows for reasoning about plans based on contradictory planning beliefs.

We bridge the two formalisms and provide a translation mechanism and a scheme for argumentation-based inference using the set-theoretic notation. The close relation to classical planning allows the exploitation of efficient state-of-the-art planning techniques, which can achieve scalable plan synthesis in complex domains. The deductive argumentation approach allows the formal specification of the problem based on standard argumentation semantics, while relying on a purely logic-based argument structure. In order to optimise the efficiency of our methods we exploit the inherent structure of the planning domain with respect to both the planning and the argumentation sub-problems.

The presented research has been conducted with practicality in mind. Given the fundamental tradeoff between representation expressiveness and reasoning tractability, we focus on a middle-ground that is expressive enough to allow succinct implementation of planning domains, but at the same time is practical, and enables reasoning with reasonably complex domains.

### 1.2.2 Research Objectives

The core purpose of this work is the formulation of practical and theoretically sound methods for dealing with MPCP problems. Our research hypothesis is outlined as follows:

> We can specify and solve the problem of multi-perspective cooperative planning based on deductive argumentation, in a way that allows the synthesis of solutions in an effective and efficient manner.

By the term deductive argumentation, we imply that every argument must have the form of a deductive proof. Effectiveness refers to the quality of the specified solutions, in terms of termination, soundness and completeness. Finally, efficiency refers to the ability of the proposed mechanisms to solve MPCP problem instances of considerable size in reasonable times.

The problem specification of MPCP and our research hypothesis raise the following questions:

- Is the problem of multi-perspective cooperative planning common?
- Is argumentation theory suitable for the specification of the MPCP problem?
- What is the quality of the proposed solution to the problem of MPCP?

The first question is related to the significance of MPCP. It examines the commonality of MPCP problem instances in multiagent domains. The second question is related to the argumentation-theoretic specification of the problem. This question can be further refined to two sub-questions. On the one hand, it inquires whether the argumentation formalism can adequately specify the notion of plan acceptability, while on the other, it refers to the expressiveness of this formalism and its ability to represent planning domains in a succinct manner. The final question refers to the notions of effectiveness and efficiency. The former describes whether the proposed methods provide sound and complete results. The latter reflects the practicality of the reasoning process, that

is, whether the proposed methods can synthesise solutions in a reasonable time in extensive domains. Based on the concrete foundation of deductive argumentation we analytically investigate important properties of our methods, including soundness and completeness, illustrating the effectiveness of the presented planning mechanisms. In order to evaluate the efficiency of our algorithms, we conduct an empirical investigation using benchmark problems from the International Planning Competition (McDermott, 2000).

### 1.2.3 Contributions

The main contributions of our work are outlined as follows:

**Formalisation of the MPCP:**

We define the problem of MPCP using two formalisms: a set-theoretic formalism adapts the classical, STRIPS-style (Fikes and Nilsson, 1972), planning notation, and a formalism based on the combination of defeasible logic programming (García and Simari, 2004) and situation calculus (McCarthy and Hayes, 1969). The first formalism deviates as little as possible from classical planning, and allows, as a result, the use of efficient automated planning techniques. The latter provides an expressive language for reasoning about contradictory dynamic domains using deductive arguments. We provide a translation mechanism, and formally show the relations between conclusions derived from theories encoded in these notations, while comparing their relative expressive power and inferential capabilities.

**Practical algorithms for reasoning and planning with MPCP domains:**

We focus on the algorithmic aspects of MPCP and discuss heuristics that exploit the inherent structure of the planning domain to prune the search space of performing derivations, generating arguments, and synthesising potential plans. Also, we formulate the problem in a format suitable for classical planners, enabling the use of highly efficient, off-the-shelf planners.

**Distributed mechanisms for reaching agreement:**

We propose a family of abstract dialogue-based collaborative protocols for distributed decision making. Different protocols are proposed designed for scenarios with different characteristics. The abstract dialogues models are concretised with respect to the problem of MPCP.

**Analytical and experimental evaluation of our methods:**
We analytically investigate important properties of our methods, such as termination, soundness and completeness, that describe their effectiveness. In addition, we experimentally evaluate the efficiency and practical relevance of our approach in benchmark planning domains form the International Planning Competition.

These contributions advance the state-of-the-art of the following research areas:

**Automated planning:** Our work is the first attempt on relaxing the assumption of classical planning that domain knowledge is consistent. The implemented planning system is the first planner capable of planning with contradictory planning beliefs in a scalable way.

**Multiagent Systems:** The main contribution to multiagent systems is the specification of an argumentation-based dialogue protocol for decentralised decision making in MPCP planning domains.

**Reasoning about dynamic domains:** We propose defeasible situation calculus, a novel formalism for reasoning about contradictory dynamic domains.

**Argumentation:** By utilising the implicit structure of the planning domain, we propose practical argumentation-based reasoning methods that allow for scalable planning in domains in which a naive argumentation approach would be infeasible.

**Artificial Intelligence:** Our work bridges research in three important areas of artificial intelligence; classical planning, reasoning about action and argumentation.

## 1.3 Thesis Structure

Our work lies at the intersection of planning, reasoning about action, argumentation and argumentation-based dialogue. Chapter 2 presents an overview of influential work from these research fields, and details methods that are closely related to the problem of multi-perspective cooperative planning. In Chapter 3, we formalise the MPCP problem and specify solution concepts based on a set-theoretic and a defeasible logic notation. The inferential results of the two formalisms are analytically related and compared. In Chapter 4, we focus on the algorithmic problems of MPCP. We exploit on

inherent characteristics of the planning domain to optimise the reasoning process, and utilise heuristic planning techniques and state-of-the-art planners to further increase the practicality of our approach. Based on argumentation-based dialogue, Chapter 5 presents a family of protocols that allow the distribution of the conflict resolution and planning tasks. Chapter 6 provides a comprehensive evaluation of our approach. We discuss our analytical results, provide an empirical investigation using benchmark planning domains, and analyse MPCP problem instances in scenarios inspired by important real-world problems. The final chapter overviews the main contributions of our approach and describes potential future research directions.

# Chapter 2

# Background

This project lies in the intersection of planning, reasoning about action, argumentation and argumentation-based dialogue. While there is limited directly related work, there is an extensive body of loosely related literature. This chapter outlines influential research that is relevant to this thesis, and presents in more detail work that is closely related to multi-perspective cooperative planning.

## 2.1 Planning

The notion of plans and the process of planning are important to MPCP. This section presents an overview of important work in automated planning, deductive planning and multiagent planning that is relevant to this thesis. In addition, we discuss in more detail problems from the planning literature that are related to MPCP and compare them to the problem of this thesis.

### 2.1.1 Automated Planning

*Planning* can be defined as the reasoning side of acting (Nau et al., 2004). *Automated planning* is the process of constructing a sequence of actions that can be applied to the current state of the world in order to achieve an objective. More specifically, a planning problem is specified according to the following:

- the state of the world
- a goal
- the actions that can be executed to affect the environment

A planner is a process that identifies a sequence of actions, which can be applied in any environment that follows this representation, and will cause a transition to a state that satisfies the specified goal. Multiple formalisms of varying expressive power have been proposed. *STRIPS* (Fikes and Nilsson, 1972) is the most influential and is the basis for the formulation of the *classical planning problem*.

#### 2.1.1.1 The Classical Planning Problem

The *classical planning problem* Weld (1999) assumes a *fully observable*, *static* and *deterministic* world. Knowledge of the environment and the specification of the available actions is complete. There are no exogenous events, and actions affect the world in a certain, deterministic manner.

The classical planning problem is defined on top of a logical language $\mathcal{L}$ containing predicate names, variable symbols, and constant symbols. $F$ is the set of literals specified by $\mathcal{L}$. This set contains the set of unground literals $F_v$ containing variables, and the set of ground literals $F_c$, grounded for every object in the language. *States* are sets of ground atoms of $\mathcal{L}$. An atom $p$ holds in a state $\sigma$ if $p \in \sigma$. A state *satisfies* a set of literals if every positive literal is part of the state and no atom which appears in a negative literal is part of the state.

**Definition 1.** *The* classical planning problem *is a tuple:*

$$P = \langle F, I, O, G \rangle,$$

*where*

- $F$ *is a set of literals,*
- $I \subseteq F_c$ *represents the initial state of the environment,*
- $O$ *is the set of planning operators, and*
- $G \subseteq F_c$ *is the goal.*

*Planning operators* are tuples of the form:

$$\langle pre, o, eff \rangle,$$

where

- $o$ is the name of the operator,

- $pre \subseteq F_v$ is the set of preconditions and
- $eff \subseteq F_v$ is a set of effects.

The sets of preconditions and effects of operator $o$ are represented as $pre(o)$ and $eff(o)$ respectively.

*Actions* are ground instances of operators. An action $a$ is applicable in a state $\sigma$ if all positive preconditions of the action is part of $\sigma$ and no negative precondition of $a$ appears in $\sigma$, formally $pre^+(a) \subseteq \sigma$ and $pre^-(a) \cap \sigma = \emptyset$. The sets $pre^+(a)$ and $pre^-(a)$ denote the positive and negative preconditions of $a$ respectively. The *state transition function* specifies how applicable actions alter the state of the world. It is represented as a function which, given the state of the environment and an action, returns the successor state after the execution of the action. The successor state is formulated by removing the negative effects of the action and adding the positive ones:

$$\gamma(a,\sigma) = (\sigma - eff^-(a)) \cup eff^+(a)) .$$

#### 2.1.1.2 Extending the Classical Planning Problem

The classical representation has been extended in various ways to increase the expressive power of the formalism and enable more concise planning theories. Syntactic extensions include typed variables, negative literals in conditions and goals, quantifiers in the conditions and effects of operators and operators with conditional effects.

Nebel (2000) formalises the notion of *expressive power* in planning formalisms and investigates how different features of propositional planning formalisms affect the planning problem. Nebel considers formalisms extending propositional STRIPS allowing combinations of the following features.

- Incomplete state specifications
- Conditional Effects
- Preconditions and effect conditions can be literals
- Formulae in preconditions and effect conditions can be boolean formulae.

The computational complexity of planning remains the same across the different formalisms. With respect to expressivity, conditional effects cannot be compiled away without causing polynomial growth of the size of the resulting plans, and that propositional formulae cannot be compiled into conditional effects and linearly preserve the plan size.

#### 2.1.1.3 Synthesising Plans

Synthesising plans is usually performed by searching the space of states or the space of plans (Nau et al., 2004). *State space planning* involves searching a graph, in which nodes represent states and edges are state transitions. In this setting, a plan is a path from the initial state to a goal state. *Forward state space search* begins from the initial state and searches the state space until a goal state is reached, whereas *backwards search* begins with the goal and generates the plan in inverse order. Backwards search selects operators that achieve subgoals and replaces subgoals with the preconditions of the operators achieving them. A plan is found if the subgoals are satisfied by the initial state. In *plan space search* nodes are partial plans and edges are plan refinement operations (Sacerdoti, 1975; Tate, 1977; Currie and Tate, 1991; McAllester and Rosenblitt, 1991; Erol et al., 1994; Nau et al., 2003).

*Planning graph* techniques (Blum and Furst, 1995, 1997) follow an alternative approach. A planning graph is arranged in layers. Odd layers are action layers (nodes correspond to action instances), whereas even layers are proposition layers (nodes correspond to propositions). Layer 0 is a proposition layer including all propositions that hold in the initial state. Edges from proposition layer nodes to action layer nodes encode preconditions, whereas edges from action layer nodes to proposition layer nodes encode effects (including maintenance actions for unaffected propositions). Action layers encode parallel actions. After the expansion of the planning graph, solution can be extracted in a backwards-chaining manner.

The extensive size of the state-space makes exhaustive search impractical. The most successful recent approaches employ heuristics guiding the search through the search space. One of the most influential heuristics (McDermott, 1996; Bonet and Geffner, 2001; Hoffmann and Nebel, 2001) is measuring the quality of a state based on the size of a plan solving a relaxed planning problem, in which delete lists (i.e. the negative effects of actions) are disregarded (making states increase monotonically after the application of actions).

*FastForward* or FF (Hoffmann and Nebel, 2001) is a highly regarded, forward state space planner, which quickly calculates the heuristic quality of states based on a planning graph approach. State space search is performed by the *enforced hill climbing* strategy: greedily move to the nearest, strictly better state discovered using breadth-first search. If enforced-hill climbing leads to a local maximum, in terms of heuristic quality, which does not achieve the goal, it fails and best-first search is invoked. Simi-

lar to enforced hill climbing, best-first search gives priority to the state with the higher heuristic quality, but also enables backtracking and guarantees finding a solution if one exists.

#### 2.1.1.4 Relevant to the Project

MPCP relaxes the implicit assumption of automated planning that planning beliefs are consistent. MPCP is strictly harder than classical planning as it imposes the additional constraint of plan acceptability on solutions. We focus on practicality and maintain a middle-ground between representation expressiveness and reasoning tractability. The set-theoretic formalism we present in Chapter 3 allows important features of standard planning representations, such as variables and conditional effects, which can be used to encode planning domains in a succinct manner. Chapter 4 presents algorithms that utilise standard heuristics and planning systems to allow efficient synthesis of plans in MPCP domains.

### 2.1.2 Deductive Planning

A deductive argumentation-based solution to MPCP requires the use of a formalism that is capable of reasoning about actions and plans in a purely deductive manner. Reasoning about action has been an major focal point of artificial intelligence research. The general idea behind this work is that all types of reasoning should be performed by a general problem solver, utilising all kinds of knowledge represented in a uniform manner.

A major obstacle in this research has been the *frame problem* (McCarthy and Hayes, 1969; Russell and Norvig, 2003). The frame problem is the problem of representing the effects of actions without having to explicitly encode what is not affected by the actions. Consider a system with $A$ actions, each one of which has $E$ effects at most. The representation of action effects requires $O(AE)$ axioms. If there are $F$ fluent predicates, then the explicit representation of what stays the same requires $O(AF)$ *frame axioms*, and typically the overall number of fluent predicates is significantly higher than the number of action effects.

Multiple action formalisms have been proposed in the literature, each offering different solutions to the frame problem. The most common formalisms are the situation calculus (McCarthy and Hayes, 1969; Reiter, 1991), event calculus (Kowalski and Sergot, 1986) and $\mathcal{A}$ (Gelfond and Lifschitz, 1993).

Apart from the frame problem other important related problems arise in reasoning about action and dynamic domains. The *qualification problem* is related to the impossibility of listing all preconditions of a real-world action. The *ramification problem* is the problem of representing all indirect, implicit effects of actions (McCarthy, 1977; Russell and Norvig, 2003) .

#### 2.1.2.1 Basic Theories of Action

Situation calculus (McCarthy and Hayes, 1969) is a language for the representation of dynamic domains. It supports three disjoint sorts: *action* represents actions; *situation* represents histories of action sequences; *object* all the rest. $S_0$ is a constant symbol representing the initial situation. The binary function symbol *do* denotes the successor situation after performing an action. *Poss* is a binary predicate symbol representing whether an action is applicable in a situation. The binary predicate symbol $\sqsubset$ defines an ordering relation over situations, where $s \sqsubset s'$ denotes that $s$ is a proper subsequence of $s'$. Symbols whose value change in different situations are called fluents, and they have an term of sort situation as their final argument.

Reasoning about dynamic domains can be performed in structured situation calculus theories, called *basic action theories*, overcoming the frame problem (Reiter, 1991, 2001). A basic action theory $\mathcal{D}$ has the following form:

$$\mathcal{D} = \Sigma \cup \mathcal{D}_{ss} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0}.$$

$\Sigma$ is a set of fundamental domain-independent axioms providing the basic properties for situations. $\mathcal{D}_{ss}$ contains a successor state axiom for each relational fluent in the domain[1] of the form $F(\vec{x}, do(a,s)) \equiv \Phi_F(\vec{x}, a, s)$, which specifies all the conditions that govern its value. The conditions under which an action can be performed are specified by the action precondition axioms which have the form $Poss(A(\vec{x}), s) \equiv \Pi_A(\vec{x}, s)$, and are included in $\mathcal{D}_{ap}$. $\mathcal{D}_{una}$ contains the unique names axioms for actions. $\mathcal{D}_{S_0}$ is a set of first-order sentences that represent the initial state of the world.

For example, consider a domain with a robot moving in a grid, picking and delivering parcels. The successor state axiom for the fluent predicate *Holds* has the following form:

$$Holds(r,p,do(a,s)) \equiv a = pickup(r,p) \vee Holds(r,p,s) \wedge a \neq deliver(r,p)\,.$$

[1] Similar axioms are also included to account for functional fluents.

The axiom states that robot $r$ holds parcel $p$, in the situation resulting from the application of action $a$ in situation $s$, if and only if either the action $a = pickup(r,x)$ is applied in $s$ or the predicate $Holds(r,p,s)$ is true in situation $s$ and $a \neq deliver(r,p)$. The first part of the body of the axiom denotes positive effects producing $p$. The second part of the axiom is the frame part and lists the conditions under which the value of the fluent persists in the successor situation. For the same domain, the action precondition axiom for the action $deliver(r,p)$ has the following form:

$$Poss(deliver(r,p,l),s) \equiv at(r,l) \wedge Holds(r,p,s)\,.$$

The action precondition axiom lists the preconditions of the action $deliver(r,p,l)$. In order to deliver the package $p$ to location $l$ in a situation $s$, $r$ must hold the $p$ and be at the delivery location.

The solution of the frame problem is based on the *causal completeness assumption* (Pednault, 1989; Reiter, 1991), which asserts that everything affecting the value of a fluent in the successor situation is accounted for in the right hand side of the relevant successor state axiom.

#### 2.1.2.2 Inference

The higher-order nature of situation calculus theories complicates inference. Regression is a powerful tool that uses the structure of basic action theories to simplify the reasoning process. The definitional form of the axioms in a basic action theory asserts that the value of a fluent in a situation is exclusively affected by the values of non-fluents and fluents in its predecessor situation. Regression exploits the definitional form of the axioms in order to simplify queries regarding future situations by substituting fluents with equivalent formulas regarding their predecessor situation, as specified in the relevant definitional axioms.

Regression can be applied to formulas with specific characteristics called *regressable formulas* (Reiter, 2001). Essentially, every situation term that is mentioned in such formulas must be rooted at $S_0$. In addition, such formulas must not quantify over situations, nor contain the symbol $\sqsubset$. Finally, they should not mention equalities between situation terms, and when the special predicate *Poss* is mentioned, it must refer to an action function symbol of $\mathcal{L}_{sitcal}$.

The *regression operator* (Reiter, 2001) eliminates *Poss* predicates by substituting them with the logically equivalent formula provided by the relevant action precondition axiom. In a similar manner, it replaces fluents regarding a situation $do(a,s)$ with an

equivalent expression uniform in $s$, as provided by the relevant successor state axiom. The operator follows this strategy until it reaches a formula consisting exclusively of initial situation fluents and non-fluent predicates and functions.

The application of the regression operator produces a query which can be answered by a subset of the theory without the foundational axioms, the successor state axioms and the action precondition axioms, thus significantly simplifying the theorem proving task.

**Theorem 1.** *(The Regression Theorem, taken from Reiter (2001)) Suppose $W$ is a regressable sentence of $\mathcal{L}_{sitcal}$ that mentions no functional fluents, and $\mathcal{D}$ is a basic theory of actions. Then, $\mathcal{D} \models W$ iff $\mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models \mathcal{R}[W]$, where $\mathcal{R}[W]$ is the formula resulting from the application of the regression operator to $W$.*

Following the previous example, we use the regression operator to answer the query $Holds(R,P,do(move(R,L),S_0))$. $R,P,L$ are ground terms representing a robot, a parcel and a location respectively.

$$\mathcal{R}[Holds(R,P,do(move(R,L),S_0))] =$$
$$\mathcal{R}[move(R,L) = pickup(R,P) \vee Holds(R,P,S_0) \wedge move(R,L) \neq deliver(R,P)] =$$
$$move(R,L) = pickup(R,P) \vee Holds(R,P,S_0) \wedge move(R,L) \neq deliver(R,P)$$

The only situation term appearing in the resulting formula is $S_0$. The original query is equivalent to the following:

$$\mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models move(R,L) = pickup(R,P) \vee Holds(R,P,S_0) \wedge move(R,L) \neq deliver(R,P)\,.$$

The unique names axioms for actions are used to simplify the resulting query with respect to the action equalities and inequalities.

#### 2.1.2.3 Synthesising Plans

Planning in a dynamic domain specified as a basic action theory can be performed through search for an *executable* situation achieving the desired goal. A situation is executable if all actions can be applied in sequence. The following abbreviation defines executability (taken from Reiter (2001)):

$$executable(s) \stackrel{def}{=} \forall a, s^*.do(a,s^*) \sqsubseteq s \supset Poss(a,s^*).$$

Using this definition, plans are defined as follows:

**Definition 2.** *Let $\mathcal{D}$ be a set of situation calculus axioms characterising some application domain, S a variable-free situation term, and $G(S)$ a situation calculus formula whose only free variable is the situation variable S. Then S is a plan for G (relative to $\mathcal{D}$) iff*

$$\mathcal{D} \models executable(S) \wedge G(S).$$

Deductive planning can be performed through proving the sentence $(\exists s).executable(s) \wedge G(s)$. Alternatively, planning can be performed by constructing ground situation terms for different sequences of actions (e.g. in a breadth first or a depth first manner) and evaluating if the plan statement holds in this situation.

#### 2.1.2.4 Implementation of Basic Action Theories

Basic action theories can be implemented as equivalent Prolog logic programs. The implementation process asserts that when a query in the logic program succeeds, then the relevant sentence is logically entailed by the theory, and whenever it fails, then the theory entails the negation of the sentence. The corresponding logic program can be obtained after a series of transformations called the Revised Lloyd-Topor Transformations (Reiter, 2001). These transform the if-halves of the definitional axioms into a syntactic form suitable for implementation as Prolog clauses. Reasoning with the resulting logic programs using Prolog's backwards chaining mechanism is equivalent to performing a series of regression steps and finally proving of the regressed formula.

#### 2.1.2.5 Relevance to the Project

Chapter 3 introduces a defeasible situation calculus variant that serves as the underlying formalism in our project. This formalism is also employed as the language in which agents communicate their beliefs about the anticipated effects of the plans when discussing about potential plans.

The defeasible situation calculus variant we will use does not offer the expressive power of the complete situation calculus language. Since we are dealing with contradictory theories, which is not the case in basic theories of action, a less expressive formalism is employed to enable tractable reasoning.

Our theories follow the structure of Reiter-style axioms. This structure enables the concise representation of both the effect and frame information associated with actions and fluents, and allows the expression of the agents' views regarding change and persistence.

The defeasible situation calculus variant is syntactically similar to logic programming implementations of basic action theories. The main distinction between such theories and our formalism is that our work enables reasoning and planning with contradictory theories.

### 2.1.3 Multiagent Planning

In multiagent systems (Wooldridge, 2002; Weiss, 1999) agents may need to coordinate in order to achieve goals that are otherwise unattainable or improve plan efficiency. Accordingly, planning agents may coordinate, formulating plans involving actions whose execution involves multiple agents. Also, agents may distribute the planning process in order to utilise all their resources and construct plans in a distributed fashion.

*Multiagent planning* is considered to be the problem of planning in the presence of multiple agents (Durfee, 1999; DesJardins et al., 1999; de Weerdt and Clement, 2009). There have been several approaches dealing with different aspects of the general problem. These can be categorised with resect to the following dimensions:

- Planning: centralised or distributed?
- Execution: single-agent or multiagent?
- Agents' attitudes towards cooperation: coopearative or strategic?
- Communication mechanism: prior to planning, during planning or during execution?
- Knowledge: shared or distributed?

In most approaches in the literature, multiagent planning usually refers to distributed planning, distributed execution, or both. The third case is the most interesting, where multiple agents are searching for a plan in parallel that involves actions contributed by multiple agents.

#### 2.1.3.1 Cooperative Distributed Problem Solving

In a general setting, multiagent planning problems involve a combination of planning and coordination problems. Some of the most influential planning approaches involve the study of coordination in multiagent systems. *Partial global planning (PGP)* (Durfee and Lesser, 1991) is one of the most influential frameworks for the coordination of

agents' activities. Agents exchange information in order to reach common conclusions about the problem-solving process. Agents individually generate partial solutions, and utilise information obtained from others to achieve non-local views of the problem. A *partial global plan* contains the overall goal and information on the agents' activities and how they should interact and exchange information to achieve the larger goal. PGP has been further refined and generalised in *Generalized PGP* to handle redundancy and coordination relationships. Other approaches view multiagent planning as the generalisation of local plans to global plans (Durfee and Lesser, 1991; Ephrati et al., 1995; Georgeff, 1983).

A different body of work in agent collaboration is based on agent theories of mental state. Cohen and Levesque (1991a,b); Levesque et al. (1990) introduce a practical model of cooperative distributed problem solving based on the concepts of *joint intentions*, *joint commitments* and *joint persistent goals*. Teamwork has been extended by several other approaches (Jennings, 1995; Wooldridge and Jennings, 1999). Wooldridge and Jennings (1999) break down the problem of cooperative distributed problem solving to the following steps:

1. Recognition of need for cooperative action
2. Team formation
3. Plan formation
4. Team action

They argue that the collective needs to come to some agreement about their joint course of action, and that such agreements can be achieved using negotiation or argumentation. Other important approaches in this field include the theory of SharedPlans (Grosz and Kraus, 1999, 1996), and STEAM (Tambe, 1997), which builds on Teamwork and SharedPlans.

#### 2.1.3.2 Cooperative Multiagent Planning

In a cooperative setting multiagent planning may involve the execution of actions by multiple agents. The extension of automated planning to account for distributed execution is one aspect of multiagent planning (Katz and Rosenschein, 1989). Complex models of distributed execution allow concurrent, interacting actions. Boutilier and Brafman (2001) modify the standard STRIPS planning problem to accommodate for

concurrent interacting actions and present a partial-order planning algorithm for centralised cooperative multiagent planning.

Contrary to single-agent planning approaches, multiagent planning approaches take advantage of the distributed nature of the problem (DesJardins et al., 1999). On the one hand, distribution of the planning problem (Corkill, 1979; Ephrati and Rosenschein, 1994a) allows the exploitation of the reasoning capabilities of multiple agents. On the other hand, by considering the multiagent nature of the problem, it is possible to break down the extensive overall planning problem to individual problems, which are potentially simpler (Lansky, 1991). The individual solutions are then merged to formulate a global plan (Stuart, 1985; Yang et al., 1992; Foulser et al., 1992; Ephrati and Rosenschein, 1993; Tsamardinos et al., 2000).

The benefits of a distributed approach are related to the degree of *coupling* (Brafman and Domshlak, 2008) in a multiagent system. Coupling reflects the amount of coordination required among the agents in the system. In domains with limited agent interaction, the distributed multiagent planning algorithms can outperform state-of-the-art centralised planners (Nissim et al., 2010).

#### 2.1.3.3 The "Classical" Multiagent Planning Problem

There is no single, universally accepted formal definition of the multiagent planning problem. Multi-Agent STRIPS (MA-STRIPS) (Brafman and Domshlak, 2008; Moses and Tennenholtz, 1995) extends the classical planning representation to account for cooperative multiagent planning.

A MA-STRIPS planning problem is the tuple $\Pi = \langle F, I, \{A_i\}_{1 \leq i \leq k}, G \rangle$.[2] $A_i$ represents the actions that agent $i$ is capable of performing. An action $a$ has the standard STRIPS syntax and semantics, $a = \langle pre, a, eff(a) \rangle$. When $k = 1$, MA-STRIPS reduces to STRIPS. The goal $G$ is shared among the agents, making the problem formulation cooperative.

MA-STRIPS does not restrict the action execution scheme. Depending on the specific approach, the agents' actions may be executed synchronously, asynchronously or in an interleaved fashion.

#### 2.1.3.4 Strategic Considerations in Planning

*Cooperative* agents share a goal and are willing to contribute towards this goal. How-

[2]The notation is slightly adapted to follow the presented single-agent planning problem formulation.

ever, autonomous agents may have *strategic* considerations in the general case that drives them towards forming coalitions and coordinating exclusively in situations in which personal gain is attainable through joint action (Ephrati and Rosenschein, 1994b; Larbi et al., 2007; Brafman et al., 2009; Jonsson and Rovatsos, 2011; Crosby and Rovatsos, 2011).

Joint plans, apart from achieving the goal, must be acceptable to the agents individually. For instance, rational autonomous agents would not subscribe to plans that achieve goals that they can reach through less costly individual plans. Strategic multiagent planning imposes additional constraints on potential solutions to planning problems. These constraints are related to the utility of the achieved goals and the cost of the actions each agent contributes to the joint plan.

#### 2.1.3.5 Communication in Multiagent Planning

Communication can be applied in different phases of multiagent coordination in order to align the agents' viewpoints or partial results obtained by the planning process (Werner, 1988; Wolverton and DesJardins, 1998). These can be categorised as follows:

- Communication before planning
- Communication during planning
- Communication during execution.

In the first case, communication is employed as a mechanism for the exchange of information related to the state of the world. Communication during the planning process is necessary when different agents construct partial plans that need to be merged (Georgeff, 1983), or when the agents need to ensure that there are no interferences between different individual plans. Communication during execution is necessary in situations in which agents need results obtained by different agents, or when problems arise during execution and re-planning is required. In this case, there is also the problem of determining which results need to be communicated and to which agents.

#### 2.1.3.6 Relevance to this project

Most multiagent planning approaches in the literature deal with the problems of coordinating the actions of multiple agents, or distributing the planning process utilising the reasoning capabilities of multiple agents. Reaching agreement on plans for action

may be hindered if different agents expect a plan to affect the environment in different ways. Such disagreements can be the result of the locality of sensing, outdated information, contradicting domain beliefs encoded by different agent designers, or simply because different agents have conducted different inferences and therefore their beliefs are not aligned. Since cautious autonomous agents will not subscribe to plans that they consider to be harmful, ineffective or questionable, methods for reaching agreement in a collaborative planning environment are needed.

This thesis focuses on the least researched dimension of multiagent planning involving the distribution of knowledge. In order to simplify the process and focus exclusively on knowledge we follow a simplified model of execution, in which actions are performed in sequence, regardless if actions are expected to be executed by single or multiple agents. We describe both centralised and distributed methods for cooperative multi-perspective planning. However, the distribution is specifically focusing on reasoning about the acceptability of the plans, while the planning process is performed individually. Finally, we focus on a cooperative setting, with agents sharing the goals. However, we consider agents to be cautious, autonomous entities that are only willing to agree on plans that seem correct with respect to their knowledge about the environment, and won't follow blindly the proposals of their peers. This imposes the additional constraint of acceptability to solutions of the planning problem, in a similar sense as in strategic multiagent planning.

### 2.1.4 Related Work in Automated Planning

This section details work from the areas of planning and reasoning about action that is related to this project. We discuss why this work is relevant and how the problems solved relate to the problem of multi-perspective planning under ontological agreement.

#### 2.1.4.1 Planning under Uncertainty

In multiagent systems, the actions of the agents make the environment dynamic. Actions performed by other agents can interfere with the plans of a planning agent, and even make them fail. Agents cannot be sure about the effects of their actions, if these are affected by the behaviour of other agents in the system. Single-agent planning in a multiagent environment can be viewed as planning under uncertainty.

The classical planning problem assumes a fully observable, deterministic environ-

ment. When these assumptions are relaxed, uncertainty is introduced (Goldman and Boddy, 1996; Boutilier et al., 1999). Uncertainty may be caused by partial observability or non-deterministic planning operators.

The planner can overcome uncertainty using test actions, which can be applied to identify the actual state of the environment, and by planning for all possible *contingencies* (Hoffmann and Brafman, 2005). Alternatively, the planner can search for a *conformant* plan (Smith and Weld, 1998; Brafman and Hoffmann, 2004; Palacios and Geffner, 2009) that can achieve the goal if it is applied in any one of the possible states the environment might be. Another alternative for dealing with uncertainty is *decision-theoretic* planning (Boutilier et al., 1999), which based on probability distributions over the outcomes of actions for every state and a preference function over outcomes, decision-theoretic planning searches for plans that maximise the excepted utility.

**2.1.4.1.1 Relevance to the Project** Our work is related to planning under uncertainty and conformant planning although the problem in this case is different. Conformant plans achieve the goal in all worlds that result from the combination of the uncertain pieces of information about the planning domain, i.e. $2^n$ for $n$ uncertain propositions. The problem we are dealing appears similar, since plans that need to satisfy $n$ different world views (one view for each planning agent). However, our system is fundamentally different from conformant and contingent planning. Our system allows persuasion, enabling the acceptance of plans that initially seemed unacceptable, on the basis of convincing for disputed propositions. As a result, if ambiguity regarding all proposition in the planning domain can be resolved, there is no uncertainty in the resulting planning problem. As a result, planning under uncertainty is complementary to our work, since its techniques can be employed if contradiction cannot be resolved and a plan that succeeds regardless of the uncertainty is required.

#### 2.1.4.2 Reasoning about Knowledge

Uncertainty can be reduced during execution through the application of *sensing actions*. Planning with sensing actions usually needs to account for all potential outcomes, and as a result (Levesque, 1996) leads to conditional plans accounting for all the possible outcomes if these actions. Planning in such settings can be performed at the knowledge level, with sensing actions treated as actions that update the agent's knowledge (Bonet and Geffner, 2000; Bertoli et al., 2001; Petrick and Bacchus, 2002).

There has been a body of work on extending reasoning about situation calculus action theories with knowledge producing actions (Scherl and Levesque, 2003; Shapiro et al., 2011; Demolombe and Parra, 2006) by incorporating specific axioms about beliefs. Reasoning about knowledge and change deals with the problems of *belief update* (belief change as a result of action), *belief expansion* (belief change as a result of new information) and in some cases the more general *belief revision*.

Knowledge producing actions can be used as the means for achieving epistemic goals. In a multiagent setting, epistemic goals aim to bring about *common knowledge* (Van Der Hoek and Wooldridge, 2002).

**2.1.4.2.1 Relevance to the Project** These approaches focus on how the beliefs of agents change after the application of actions (including knowledge producing actions). The use of these methods in a multiagent setting can provide information about the knowledge of every agent related to the application of different plans. Therefore, they can be used to identify plans for which the knowledge of the agents is aligned. However, they do not offer ways to align the agents' views of the world themselves.

#### 2.1.4.3 Revising Action Theories

In dynamic environments agents may hold outdated or erroneous action theories. Research in action theory change (Eiter et al., 2010; Varzinczak, 2010) involves methods for revising action theories in the light of new information.

Eiter et al. (2010) defines the problem of *Action Description Update* in the context of the action language $\mathcal{C}$ (Giunchiglia and Lifschitz, 1998). The input of the problem is an action description $D$, with unmodifiable $D_u$ and modifiable parts $D_m$, the update that must be incorporated $I$, a set of "hard" and "soft" constraints $C = C_o \cup C_p$, and a preference relation over action descriptions $\sqsubseteq_c$. An action description $D'$ is a solution to the action description problem if it accomplishes the update of $D$ by $I$ relative to $C$ under the following conditions:

- $D'$ is consistent.
- $D'$ contains the new information $I$ and the unmodifiable knowledge $D_u$, and the causal laws from the modifiable knowledge $D_m$ are either accepted or disposed.
- The constraints $C_o$ are imposed on $D'$.

- There is no action description $D''$, for which the previous hold, such that $D' \sqsubseteq_c D''$.

Varzinczak (2010) focuses on action theory change in the context of the multimodal logic $K_n$ (Popkorn, 1994). This method also focuses on constrained-based updates, that is the resulting theory must respect a set of laws. Action theories contain static laws describing constraints, effect laws describing action effects and executability laws describe action applicability. *Contraction* of static, effect and executability laws is performed based on minimal change based on the notion of distance between Kripke-models.

**2.1.4.3.1 Relevance to the Project** This work is complimentary to our approach. In our approach, agents do not have to revise their theories to account for axioms communicated by the other parties. However, they are forced to accept arguments that they cannot defeat. Accepting such arguments entails acceptance of their conclusions. Resolution of contradictory views is situation dependant. Resolution of contradictions is based on operator specifications but also depends on the relevant conditions. In different situations with different conditions, different resolution results may be obtained. As a result, it is not always possible to prioritise which beliefs must be used to update the theory.

Our methods do not exclude the option of the agents individually revising their action theories in light of new information. However, these methods cannot be applied to solve the multi-perspective cooperative planning problem, since it is not clear which views should be prioritised, triggering updates. Additionally, it is not clear if the specification of the employed formalisms are adequate for the representation of classical planning domains without additional constraints.

In order to provide an overall solution to our problem without the use of argumentation, belief revision methods may be used to merge the beliefs of multiple agents and generate a shared theory by removing the inconsistencies. Such a process would have to be centralised. Also, revising beliefs prior to planning requires the resolution of all conflicts, even those completely unrelated to concrete potential plans.

This is closely related to the main distinction between the use of argumentation and belief revision, the social and cognitive side of epistemic reasoning (Paglieri and Castelfranchi, 2005). An argumentation-based method is better suited for a decentralised setting, since it focuses on persuasion regarding specific conflicts that are relevant to a concrete plan. Also, argumentation mechanisms (even if conducted in a

centralised fashion) generate justifications that can be used to explain and persuade autonomous agents towards accepting a plan. Falappa et al. (2009) discuss the relevance of argumentation and belief revision in more detail.

#### 2.1.4.4 Defeasibility in Planning and Reasoning about Action

Pollock (1987, 1998) emphasises the need for defeasible epistemic reasoning for planning agents in complex environments, where information about the world is uncertain and possibly incomplete. The author argues that the product of the planning process is bound to the assumptions of the planning knowledge, and since in such complex, rapidly changing environments epistemic knowledge is in principle defeasible, plans should also be. The author presents a regression-based planning approach based on the defeasible reasoner OSCAR.

There has been a body of work related to handling defeasibility in action domains (Baral and Lobo, 1997; Zhang, 2003). Baral and Lobo (1997) extend the language $\mathcal{A}$ and incorporate defeasible constraints and effect propositions, using an extended logic programming approach. Zhang (2003) introduces action languages $\mathcal{AT}^0$, $\mathcal{AT}^1$ and $\mathcal{AT}^2$ handling defeasible action constraints, defeasible observations and defeasible and abnormal effects of actions. Their approach is based on prioritised logic programs.

**2.1.4.4.1 Relevance to the Project** Contrary to these approaches we do not focus on reasoning with theories in which action is represented in a defeasible manner. Our focus is on resolving contradictions across the theories of multiple agents. Therefore, the defeasibility in our problem lies in the collective beliefs the agents hold. Our representation is based on an implementation of situation calculus basic action theories in defeasible logic programming. Accordingly, we use successor state axioms to represent the effects of actions. We discuss this choice further in Section 6.1.2.5 and compare it against an alternative formulation based on defeasible reasoning, which is inspired from Zhang (2003). A representation based on successor state axioms provides a more succinct representation, and allows the resolution of contradicting views about frame axioms as well as action effects.

#### 2.1.4.5 Summary

There is extensive work in automated planning and reasoning about action that is relevant to multi-perspective cooperative planning. However, no work in these fields deals

with the problem of this thesis. MPCP is concerned with ambiguity, which compared to uncertainty, can potentially be resolved based on relevant information held by the agents. This makes the nature of our approach defeasible. However, compared to planning with defeasible operators, we do not use defeasibility to reason about default persistence, but employ argumentation-based methods to resolve contradictory inferences made from the agents' theories. The resolution of such conflicts allows the agents to agree on conclusions, but cannot indicate which beliefs and operator specifications should be preferred over others in the general case, making the use of prioritising belief revision methods infeasible.

## 2.2 Argumentation

In order to formalise MPCP, we need to concretise the notion of plan acceptability. This notion determines when it is rational to accept a plan if there exist different views about the planning domain. The proposed formalisation is base on argumentation theory.

Argumentation (Prakken and Vreeswijk, 2002; Besnard and Hunter, 2008; Rahwan and Simari, 2009) is a mechanism for conflict resolution. Argumentation theory provides methods for the evaluation of the acceptability of arguments. These methods can be used by a single agent, or combined with a dialogue process, enable multiple agents to present their beliefs in the form of arguments. Argumentation provides a rich form of communication, since it enables agents not only to present claims, but also provide justifications for their beliefs. In light of new information, or better justification, agents can identify acceptable beliefs and alter their view of the world.

### 2.2.1 Abstract Argumentation

Dung views argumentation at an *abstract level* (Dung, 1995; Baroni and Giacomin, 2009). Individual arguments are treated as abstract entities, disregarding their internal structure and meaning. Conflict between arguments is represented by the binary *attacks* relation. A set of abstract arguments and the attacks among them form an argumentation system.

**Definition 3.** *An* argumentation framework *is a structure* $AF = \langle Args, Attacks \rangle$ *where* $Args$ *is a set of arguments and* $Attacks \subseteq Args \times Args$ *is a binary* attacks *relation between arguments.*

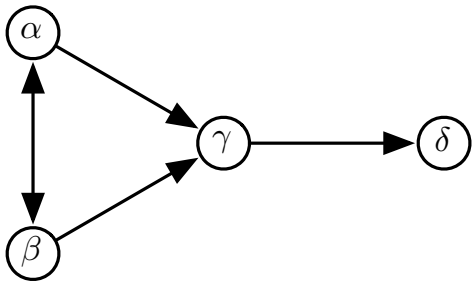



Figure 2.1: Argumentation graph depicting the argumentation framework $AF_1 = \langle\{\alpha,\beta,\gamma,\delta\},\{\alpha\rightarrow\beta,\alpha\rightarrow\gamma,\beta\rightarrow\alpha,\beta\rightarrow\gamma,\gamma\rightarrow\delta\}\rangle$

An attack between two arguments $\alpha$ and $\beta$ is denoted as $\alpha\rightarrow\beta$. Also, the attack relation is generalised for sets of arguments. A set of arguments $S$ is said to attack an argument $\beta$ if there exists an argument $\alpha \in S$ such that $\alpha\rightarrow\beta$.

Consider for example the following argumentation framework:

$$AF_1 = \langle\{\alpha,\beta,\gamma,\delta\},\{\alpha\rightarrow\beta,\alpha\rightarrow\gamma,\beta\rightarrow\alpha,\beta\rightarrow\gamma,\gamma\rightarrow\delta\}\rangle.$$

Figure 2.1 depicts $AF_1$ in the form of a directed graph, called an *argumentation graph*. Nodes correspond to arguments, whereas edges represent defeats.

If the arguments in an argumentation framework do not form a conflict-free set, then it is not rational to accept all of them in the set at the same time. *Argumentation semantics* formally define ways to evaluate the acceptability of the arguments in an argumentation framework. *Extension-based argumentation semantics* specify sets of arguments that are collectively acceptable. These sets are called *extensions* of the argumentation framework.

Multiple extension-based argumentation semantics have been proposed in the literature (Dung, 1995; Baroni and Giacomin, 2007). Before we introduce the *preferred* and the *grounded* semantics, we introduce some useful definitions.

**Definition 4.** *Given an argumentation framework $AF = \langle Args, Attacks\rangle$, a set of arguments $S \subseteq Args$ is* conflict-free *if and only if there do not exist any arguments $\alpha,\beta \in S$ such that $\alpha\rightarrow\beta$.*

Here, conflict-freeness simply means that no two arguments in a subset of *Args* attack each other.

**Definition 5.** *Consider the argumentation framework $AF = \langle Args, Attacks\rangle$. An argument $\alpha \in Args$ is* acceptable with respect to a set $S \subseteq Args$ *of arguments if and only if for all $\beta \in Args$ such that $\beta\rightarrow\alpha$, there exists $\gamma \in S$ such that $\gamma\rightarrow\beta$.*

Acceptability of an argument with respect to a set $S$ means that if there is any other attackers of the argument, some element of $S$ will attack this attacker.

**Definition 6.** *Given an argumentation framework $AF = \langle Args, Attacks \rangle$, a set of arguments $S \subseteq Args$ is called* admissible *if it is conflict-free and if each argument in $S$ is acceptable with respect to $S$.*

Admissibility, often taken as a criterion to determine whether a set of arguments can be reasonably maintained, requires that every member of the set is acceptable with respect to the set, and that this set is conflict-free.

#### 2.2.1.1 Preferred Semantics

Preferred semantics are based on the notion of the *preferred extension*, which refers to the maximal, conflict-free sets of arguments that defend themselves against every attack (i.e. attack all their attackers).

**Definition 7.** *A* preferred extension *of an argumentation framework is a maximal (with respect to set inclusion) admissible set of the argumentation framework*

*Credulous preferred semantics*, require for an argument to be part of at least one preferred extension to be acceptable overall. *Sceptical preferred semantics* require that the argument is contained in all preferred extensions of the argumentation framework.

Following on the argumentation framework $AF_1$ from Figure 2.1, there are two preferred extensions: $\{\alpha, \delta\}$ and $\{\beta, \delta\}$. A credulous agent will accept arguments $\alpha, \beta, \delta$, whereas a sceptical agent will only accept $\delta$. The rationality behind sceptical preferred semantics is that regardless of which preferred extension is selected, and in this case regardless of either selecting $\alpha$ or $\beta$ (given that both cannot hold at the same time), $\gamma$ is defeated. As a result $\delta$ can be accepted.

#### 2.2.1.2 Grounded Semantics

Grounded (sceptical) semantics are defined using the *characteristic function* of an argumentation framework.

**Definition 8.** *The* characteristic function *of an argumentation framework $AF = \langle Args, Attacks \rangle$, is the function $\mathcal{F}_{AF} : 2^{Args} \rightarrow 2^{Args}$, which is defined as follows:*

$$\mathcal{F}_{AF}(S) = \{\alpha \mid \alpha \textit{ is acceptable } w.r.t.\ S\}.$$

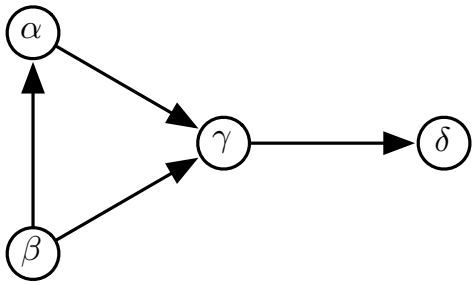



Figure 2.2: Argumentation graph depicting the argumentation framework $AF_2 = \langle\{\alpha,\beta,\gamma,\delta\},\{\alpha\rightarrow\gamma,\beta\rightarrow\alpha,\beta\rightarrow\gamma,\gamma\rightarrow\delta\}\rangle$

**Definition 9.** *The grounded extension of an argumentation framework AF, denoted by $GE_{AF}$ is the least fixed point of $\mathcal{F}_{AF}$.*

Consider the sequence:

$$\mathcal{F}_{AF}^{0} = \emptyset$$

$$\ldots$$

$$\mathcal{F}_{AF}^{i+1} = \{\alpha \in Args \mid \alpha \text{ is acceptable with respect to } \mathcal{F}_{AF}^{i}\}$$

$$\ldots$$

According to Dung (1995), it holds that: all arguments in $\cup_{i=0}^{\infty}(\mathcal{F}_{AF}^{i})$ are in $GE_{AF}$, and if each argument is attacked by at most a finite number of arguments, then an argument is in $GE_{AF}$ if and only if it is in $\cup_{i=0}^{\infty}(\mathcal{F}_{AF}^{i})$.

Figure 2.2 depicts the following argumentation framework:

$$AF_2 = \langle\{\alpha,\beta,\gamma,\delta\},\{\alpha\rightarrow\gamma,\beta\rightarrow\alpha,\beta\rightarrow\gamma,\gamma\rightarrow\delta\}\rangle.$$

$\mathcal{F}_{AF_2}^{1}$ contains $\beta$, the only argument that is not attacked. $\mathcal{F}_{AF_2}^{2} = \{\beta,\delta\}$, since $\{\beta\}$ attacks the attackers of $\delta$. This is the least fixed point of the characteristic function since $\mathcal{F}_{AF_2}^{3} = \mathcal{F}_{AF_2}^{2}$. Therefore, the arguments in $AF_2$ that are justified with respect to grounded semantics are $\{\beta,\delta\}$. The argumentation framework $AF_1$ does not have any arguments that are acceptable with respect to the grounded argumentation semantics.

#### 2.2.1.3 Preferences over Arguments

The previous examples illustrate that in certain cases mutual attacks appear in argumentation frameworks. Such ties may be broken using additional information about the preference of these arguments. This is usually represented in the form of an ordering among the arguments in an argumentation framework. Intuitively, preference

orderings may be based on information regarding the credibility or source of arguments, or their internal structure.

*Preference-based argumentation* (Amgoud and Cayrol, 1998) introduces a preference ordering among arguments that can be useful to break mutual attacks. Accordingly, a preference-based argumentation framework is a triple $\langle Args, Attacks, Pref \rangle$, where $Pref$ is a partial preordering of the arguments in $Args$.

### 2.2.2 Proof Theories for Abstract Argumentation

The definition of argumentation semantics provides a formal specification of the notion of acceptability. This section discusses the most commonly used algorithms for the evaluation of argument acceptability. An overview of proof theoretic methods for abstract argumentation can be found in Modgil and Caminada (2009).

#### 2.2.2.1 Labelling

One way to compute the acceptability of arguments is to assign labels to the arguments in the argumentation graph. *Labelling* is an assignment of a label from the set $\{IN, OUT, UNDEC\}$ (for undecided) to every argument in the argumentation framework. *Reinstatement* labellings (Caminada, 2006) abide to the following principles:

- At least one attacker for an argument labelled $OUT$ must be labelled $IN$.
- All attackers for an argument labelled $IN$ must be labelled $OUT$.

Multiple such labellings may exist for an argumentation framework. Additional restrictions specify compliance to different argumentation semantics. For instance, labellings with maximal sets of arguments labelled $IN$ or $OUT$ correspond to preferred semantics, whereas labellings with minimal sets of of arguments labelled $IN$ or $OUT$ or maximal sets of $UNDEC$ arguments are connected to grounded abstract argumentation semantics.

#### 2.2.2.2 Argument Games

*Argument games* (Vreeswijk and Prakken, 2000; Dunne and Bench-Capon, 2003) evaluate the acceptability of arguments through a dialectical process between two parties called a *dispute*. One party plays the role of the *proponent*, leaving the role of *opponent* to the other party. In the single agent case, the agent assumes both roles. The role

of the proponent is to introduce arguments defending the argument that is evaluated, while the opponent is attempting to attack this argument and the arguments defending it.

The game is initiated with the proponent introducing the argument to be evaluated. Then it is the opponent's turn to move an argument attacking the argument presented by the proponent (if such an argument exists). The game progresses with the players exchanging turns and introducing arguments attacking the previous argument of the other party.

Throughout this process, sequences of attacks are generated, which are called *dispute lines*. A dispute line consists of arguments defending the evaluated argument (line of defence) and arguments attacking these arguments (line of attack). Since there may exist multiple attackers for an argument, there may exist multiple different moves for a player. The collection of all possible dispute lines form a *dispute tree*.

The requirement that every argument in a dispute line must attack the previously introduced argument is not sufficient to produce a correct reasoning scheme. It does not safeguard against circular attacks, causing the construction of dispute lines of infinite length, or fallacies in the line of reasoning, as for instance lines of defence containing conflicting arguments. In order to safeguard against fallacies and to guarantee correct results, specific rules must govern the legality of moves in the argument game. Such rules have been shown to produce correct results with respect to different argumentation semantics (Vreeswijk and Prakken, 2000; Dunne and Bench-Capon, 2003).

### 2.2.3 Deductive Argumentation

In order to provide a concrete argumentation-based reasoning system, abstract argumentation is not sufficient, unless it can be combined with a mechanism that generates arguments and calculates the attacks relations among them. In logic-based argumentation, the internal structure of arguments is defined based on an inference procedure in a knowledge base.

Prakken and Vreeswijk (2002) overview different logics that have been proposed for deductive argumentation-based reasoning. According to their analysis the main elements of deductive argumentation systems are:

- An underlying logical language.
- Definitions of an argument, conflicts between arguments and the attack relation.

- Definition of the assessment of arguments, defining defeasible logical consequence.

Examples of concrete argumentation systems are assumption-based argumentation (Bondarenko et al., 1997; Dung et al., 2009), defeasible logic programming (García and Simari, 2004) and argumentation based on classical logic (Besnard and Hunter, 2001). Our system follows defeasible logic programming.

### 2.2.4 Defeasible Logic Programming

*Defeasible Logic Programming* (DeLP) (García and Simari, 2004) is based on a combination of logic programming and defeasible argumentation. The following analysis follows García and Simari (2004).

The DeLP language is defined in terms of *facts*, *strict rules* and *defeasible rules*. Facts are positive or negative literals. Literals are ground atoms $A$ or negated ground atoms $\sim A$. The symbol "$\sim$" represents strong negation. Strict rules encode non-defeasible knowledge. They have the following form:

$$L_0 \leftarrow L_1, L_2, \ldots, L_n,$$

where $L_0, L_1, \ldots, L_n$ are ground literals and $n \geq 0$. Accordingly, defeasible rules are represented as follows:

$$L_0 \prec\!\!\!- L_1, L_2, \ldots, L_n.$$

$L_0, L_1, \ldots, L_n$ are ground literals and $n \geq 0$. Defeasible rules describe that if there are reasons to believe that the body of a rule holds, then there are reasons to believe that the head holds as well. Defeasible rules with an empty body are called *presumptions*.

The defeasible implication symbol distinguishes defeasible rules, which are interpreted as "tentative information that may be used if nothing can be posed against it". For instance, the non-defeasible knowledge that all penguins are birds is represented using a strict rule:

$$bird \leftarrow penguin.$$

On the contrary, the knowledge that birds are usually able to fly is encoded using the defeasible rule:

$$flies \prec\!\!\!- bird.$$

Both symbols "$\leftarrow$" and "$\prec\!\!\!-$" denote a meta-relation between literals without contraposition.

A *defeasible logic program* $\mathcal{P} = (\Pi, \Delta)$ is a possible infinite set of facts, strict rules $\Pi$ and defeasible rules $\Delta$. Defeasible logic programs are ground. 'Schematic rules" with variables (Lifschitz, 1996) are sometimes used to represent all ground instances of these rules.

Ground defeasible rules can be used in sequence to create inference chains called *defeasible derivations*. A defeasible derivation is a finite sequence $L_1, L_2, \ldots, L_n = L$ of ground literals. For each literal $L'$ in the sequence there exists a rule $R$ with $Head(R) = L'$, and all literals appearing in its body appear in sequence before $L'$. A defeasible derivation represents reasons to believe the derived statements, without meaning that these statements are necessarily true, since contradicting statements may be defeasible derived. If the rules used for a derivation are exclusively strict rules, the derivation is called a *strict derivation*.

The derivations provide the basis for the construction of arguments. Given a defeasible logic program $\mathcal{P} = (\Pi, \Delta)$, an *argument structure* is a tuple $\langle \mathcal{A}, h \rangle$, where $h$ is a literal and $\mathcal{A} \subseteq \Delta$ such that:

1. there exists a defeasible derivation for $h$ from $\Pi \cup \mathcal{A}$
2. $\Pi \cup \mathcal{A}$ is *non-contradictory*, and
3. $\mathcal{A}$ is *minimal*, i.e. there is no proper subset of $\mathcal{A}$ satisfying both (1) and (2).

The literal $h$ is called the *conclusion*, or the *claim*, of the argument. $\mathcal{A}$ is called the support. Arguments indicate reasons towards a claim. They are minimal, non-contradictory sets of rules that defeasibly infer a claim. A set of rules is non-contradictory if there exists no literal which can be defeasibly inferred from the set, if its complement can be also inferred from the same set. An argument $\langle \mathcal{A}', h' \rangle$ is a *sub-argument* of $\langle \mathcal{A}, h \rangle$ if $\mathcal{A}' \subseteq \mathcal{A}$.

The attack relation between arguments is specified using the notion of *disagreement*. Given a defeasible logic program $\mathcal{P} = (\Pi, \Delta)$, two literals $h_1$ and $h_2$ disagree if the set $\Pi \cup \{h_1, h_2\}$ is contradictory. An argument $\langle \mathcal{A}_1, h_1 \rangle$ attacks another argument $\langle \mathcal{A}_2, h_2 \rangle$ if there exists a sub-argument $\langle \mathcal{A}_2', h_2' \rangle$ of $\langle \mathcal{A}_2, h_2 \rangle$ such that $h_1$ and $h_2$ disagree.

Arguments can be prioritised so as to resolve disagreements. One way to determine argument priorities is based on the *generalised specificity principle* (Poole, 1985; Simari and Loui, 1992): higher preference is assigned to arguments with greater information content or more restricted use of rules. Another alternative is prioritisation based on rule priorities. In this case, the preference of arguments is calculated based on the least preferred rule in their support.

The notion of *defeat* is based on the attack relation and the arguments' preference ordering. An argument $\langle \mathcal{A}_1, h_1 \rangle$ is a proper defeater of another argument $\langle \mathcal{A}_2, h_2 \rangle$ if it attacks $\langle \mathcal{A}_2, h_2 \rangle$ at a sub-argument $\langle \mathcal{A}'_2, h'_2 \rangle$ and $\langle \mathcal{A}_1, h_1 \rangle$ has higher preference over $\langle \mathcal{A}'_2, h'_2 \rangle$. If these arguments are equally preferred then $\langle \mathcal{A}_1, h_1 \rangle$ is called a *blocking defeater*.

Belief acceptability in DeLP is defined using the notion of warrant based on *dialectical analysis*. A query $q$ is *warranted* if an argument supporting $q$ is found undefeated by the warrant procedure. The warrant procedure is based on the generation of a *dialectical tree*. The root of this tree is the argument supporting the query. The tree is expanded by considering the defeaters of this argument, their defeaters and so on. Every line from the root of the tree to a leaf is called an *argumentation line*. Constraints assert that the generation of infinite argumentation lines is avoided.

DeLP has been extended to support the *default negated* literals. Accordingly, *Extended defeasible rules* allow default negation "*not*" in the body of a rule. For instance, consider the following example attributed to John McCarthy (Gelfond and Lifschitz, 1990):

$$\sim cross\_railway\_tracks \prec not \sim train\_is\_coming.$$

This statement represents that there are reasons to believe that we should not cross the railway tracks, if there is absence of sufficient evidence to believe that the train is not coming. Default negated literals are treated as assumptions.

The definitions of defeasible derivations, argument and defeat are also adapted to account for default negated literals. Defeasible derivations ignore literals preceded by default negation, since these are treated as assumptions. Defeaters may attack these assumptions regardless of their preference.

### 2.2.5 Dialogical Argumentation Systems

The inherent dialectical nature of argumentation has been investigated in the form of dialogical argumentation systems. In a distributed, multiagent setting, argumentation can serve as a mechanism for reconciling conflicts among the agents. McBurney and Parsons (2002), Prakken (2006) and McBurney and Parsons (2009) overview dialogue games for agent argumentation.

Dialogue systems can be categorised with respect to their purpose. This is called the *dialogue goal* and describes the reason behind the dialogue. Walton and Krabbe (1995) provide a taxonomy of dialogues with respect to their purpose:

- persuasion
- negotiation
- information seeking
- deliberation
- inquiry
- quarrel

An important element in argumentation-based dialogue is the communication language and the protocol. Some *communication language* is necessary to specify the locutions the agents can exchange. The *dialogue protocol* specifies how from the history of the moves made in the dialogue, the possible legal moves for the next iteration can be computed. The protocol accounts for commencement and termination rules and the outcome of the dialogue. In addition, the dialogue protocol specifies the conditions and effects governing the agents' dialogue moves with respect to the state of the dialogue. The state of the dialogue is determined by the already exchanged messages and the contents of the agents' commitment stores.

*Commitment stores* capture commitments (Hamblin, 1970) the agents made during the dialogue. Commitment stores allow for the verification of the consistency of the agents' statements, even though it is impossible to validate if these reflect their actual beliefs.

The topic language is the language employed for discussion about the domain. It is considered to be shared among agents. This assumption is necessary in order to ensure that the information agents exchange is meaningful. Sharing the topic language is similar to sharing the ontology for the domain in question. The context for the dialogue is a subset of the topic language that the agents consider to be common knowledge before the beginning of the dialogue.

### 2.2.6 Relevance to the Project

Argumentation theory provides the means for the specification of the notion of acceptability in the light of contradictory information. We utilise standard argumentation theory definitions for specifying formal solution concepts for the problem of multi-perspective cooperative planning.

The use of argumentation-theoretic notions in combination with a defeasible language for representing dynamic domains enables the representation of the agents' views about plans and their anticipated effects in a structured, purely logical manner.

Argumentation-based dialogue enables distribution of the reasoning process to multiple agents. Chapter 5 presents a dialogue protocol that exploits this, providing a distributed solution to the problem of multi-perspective cooperative planning under ontological agreement.

### 2.2.7 Related Work in Argumentation

This section overviews the most relevant work from the area of argumentation to the problem of multi-perspective cooperative planning.

#### 2.2.7.1 Argumentation-based Negotiation for Coordination

Argumentation mechanisms have been proposed for the coordination of the activities of agents. These works utilise the conflict-resolution capabilities of argumentation, providing mechanisms for the coordination of autonomous agents with individual goals and interdependencies in their plans for action. This section presents approaches in argumentation-based negotiation for coordination and discusses their relevance to this thesis.

Sycara (1989) argue that persuasive argumentation can be employed in non fully cooperative multiagent environments in order to increase cooperativeness and bring about convergence to a global solution. The PERSUADER system is described in the domain of Labour mediation. A persuader agent employs persuasive argumentation in order to influence the beliefs of a persuadee and increase the potential of cooperation.

Kraus et al. (1998) follow up this line of work, developing a formal logical model of the mental states of agents based on Beliefs, Desires and Intentions (BDI), and use argumentation as a mechanism for persuasion. Cooperation is achieved by influencing change in other agents' intentions, using arguments expressing threats, promises of future reward and appeals to past reward, precedents, prevailing practice or self-interest.

Parsons et al. (1998) present a formal model of argumentation-based reasoning and negotiation for autonomous agents. The authors show how this model can be used to coordinate negotiating BDI agents.

Tambe and Jung (1999) utilise argumentation to resolve conflicts in the beliefs and plans for action of the individual agents participating on team action. They present

CONSA (COllaborative Negotiation System based on Argumentation), a system relevant for teamwork models (Tambe, 1997) capable of dealing with conflicts regarding beliefs about jointly initiating or terminating team operators or conflicts about individual operators and roles. This system in based on Toulmin's argument model Toulmin (1958).

Clement et al. (2004) present SHAC (Shared Activity Coordination), an argumentation-based system for the negotiation of the shared activities of agents. SHAC provides an algorithm for interleaving planning and sharing of plan information with respect to shared activities.

**2.2.7.1.1 Relevance to the Project** Argumentation-based approaches for the coordination of shared activities are relevant to our work, especially with respect to the use of augmentation for the resolution of conflicts among individual plans in a multiagent environment. However, the focus is different. We do assume that agents are cooperative (share goals) and do not have private, individual goals. In addition, we focus on agents that have different models of the environment, both in terms of the state of the world and the specification of actions. Finally, since our model allows the specification of and reasoning about concrete sequential plans, which may involve execution by multiple agents, we do not account for conflicts arising from individual plans.

#### 2.2.7.2 Argumentation-Based Practical Reasoning

Practical reasoning can be viewed as a two-step process involving *deliberation* and *means-end reasoning*. Deliberation is concerned with reasoning about the desires of the agent and goal-settings, whereas means-end reasoning deals with reasoning about actions and plans that can achieve these goals.

Argumentation-based practical reasoning employs the conflict resolution capabilities of argumentation theory to reconcile conflicts between beliefs, intensions, desires. Different approaches deal with these aspects with respect to a single agent's internal reasoning process, or as a multiagent process for coordinating the activities of multiple agents.

This section reviews the works in argumentation-based practical reasoning. We also discuss the relevance these projects have to the problem of multi-perspective multiagent planning.

Amgoud and Cayrol (2004) and Amgoud (2003) present a framework that computes consistent sets of intentions from a potentially conflicting sets of desired and

a set of beliefs. Reasoning is performed using rules specifying *planing rules* of the form: $\phi_1 \wedge \phi_2 \wedge \ldots \phi_n \rightarrow h$, specifying that if the agent achieves $\phi_1 \wedge \phi_2 \wedge \ldots \phi_n$, then it is possible to achieve $h$. Consistent intentions contain desires that can be achieved by consistent *complete plans*, which include all actions necessary to achieve a given desire.

Hulstijn and van der Torre (2004) propose a framework for dealing with goals and plans. Compared to precious work, the authors agree that conflicts between plans are fundamentally different to conflicts about beliefs usually studied in argumentation theory. However, they present an approach for reasoning with standard Dung-style abstract argumentation frameworks.

Rahwan and Amgoud (2006) present an argumentation framework that facilitates argumentation about beliefs, generation of consistent desires and construction of plans that can achieve these desires. Utilities measuring the worth of the desires and the cost of the required resources are employed in order to evaluate the strength of the related arguments. Reasoning is performed on top of three interacting argumentation frameworks: one for beliefs, one for desires and one for plans. Argumentation over beliefs follows Dung's framework (Dung, 1995) as it was extended by Amgoud and Cayrol (2002) to incorporate the strength of arguments. Desires are supported by *explanatory* arguments describing the beliefs and other desires that justify them. Such arguments may be defeated either by belief arguments undercutting the beliefs that support the desire, or by other desire arguments undermining the desirability of the desires. Plans are represented by *instrumental arguments*, which use *planning rules* to explain the *achievability* of desires. A planning rule about a desire expresses that if a set of desires is achieved and a set of resources is used then this desired is achieved. The acceptability of arguments is calculated, and *intention sets* are formed for desires that can be achieved together.

Rotstein and Garcıa (2006) propose an argumentation-based framework for reasoning about agent beliefs and desires within a single argumentation system. Agents' beliefs and filtering rules are defined as a defeasible logic program using facts and defeasible rules. Rotstein et al. (2007, 2008) extend this work introducing agent intentions, and generalise the filtering process to account for different agent attitudes (accepting warranted desires, or desires whose complement is not warranted). Intention rules specify the preconditions and constraints under which filtered desires can be selected to pursue.

Amgoud et al. (2008, 2011) argues that the separation of practical reasoning into

two distinct processes may result in the selection of infeasible desires, even if feasible alternatives exist. This work extends previous work introducing one-step generation of intentions. In this approach, plans are considered ways of achieving a desire. They are specified by their preconditions, their effects and the desire that is reached by the plan.

Atkinson (2005) describes an *argument scheme* following Walton's *sufficient condition scheme for practical reasoning* (Walton, 1996).

> In the current circumstances $R$, we should perform action $A$, that will realise goal $G$, which will promote some value $V$.

Justifications about action are instantiations of this scheme. Challenges to these justifications are identified through a series of *critical questions*. Examples of such questions are the following:

- Are the believed circumstances true?
- Assuming the circumstances, does the action have the stated consequences?
- Is the action possible?
- Assuming the circumstances and that the action has the stated consequences, will the action bring about the desired goal?
- Does the goal realise the value stated?

These critical questions account for resolution of disagreements related to *problem formulation*, *epistemic reasoning* and *action selection*. Values are employed as qualitative reasons explaining the significance of certain affairs. Atkinson et al. (2005) present a dialogue protocol for argumentation over proposals for actions. In addition, Atkinson and Bench-Capon (2007a,b) formalise the argument generation process by grounding the underlying model to an *action-based alternating transition system*.

Medellin-Gasque et al. (2011) extend the scheme for practical reasoning to account for plans and the temporal aspects arising from the combination of actions. The resulting scheme accounts for a comprehensive collection of 66, informal for the most part, critical questions.

Tang and Parsons (2005) present an argumentation-based approach to deliberation, in terms of goal selection, reduction of this goal to sub-goals, and formation of a plan to achieve the overall goal. The environment is described in terms of states, and actions causing transitions among states. The authors refer to transitions caused by actions, or plans, from initial states to goal states as *nisi*. The deliberation process is initiated with

a nisus, from the initial state to the goal state, and is conducted in phases. In each phase, the agent decomposes intermediate nisi using a set of possible partial plans. Plans that achieve the intermediate nisi are merged to form a plan for the initial transition. The deliberation process is combined with argumentation, enabling reasoning about the acceptability of plans in single-agent deliberation. Also, the authors present a two-party deliberation dialogue protocol enabling the distribution of the process.

Toniolo et al. (2011) present a series of argument schemes that can be used for argumentation-based deliberative dialogues for collaborative planning. This work focuses on conflicts among the plans of agents, which may be caused by concurent actions, plan constraints or norms the agents must adhere to. The model is formalised using Reiter-style situation calculus extended to account for norms and durative actions.

**2.2.7.2.1 Relevance to this project** Most of the aforementioned work focus primarily on the deliberation problem, dealing with the relations among beliefs, plans and desires. We are interested on the means-end planning aspects of the problem of practical reasoning, when the agents' perspectives contain contradictory information.

In our work the term plan is closely related to the standard notion of a classical plan. On the contrary, most of the proposed works refer to monolithic plans (similar to single step actions), or employ non-standard notions, different from classical AI planning. In the case of monolithic actions, the problem is different, with the focus being on finding sets that are consistent, in the sense that are applicable together, and achieve agent desires. On the contrary, classical planning treats plans as sequences of actions that can be applied to affect the environment.

In addition, although these frameworks are expressive with respect to being able to represent the beliefs, desires and plans, they are restrictive with respect to the representation of actions. Their propositional nature does not allow concise representations of realistic planning domains.

The underlying formalisation is very important. Some relevant approaches represent the agent's domain knowledge as a state transition system. Usually, states are monolithic and actions are simply transitions in the system. It is not clear how such a formalism can facilitate the aggregation of agents' views. It is not clear how one can partially agree with some aspects of the operator, while at the same time disagreeing with others. Additionally, it is not clear how such mechanisms will treat situations in which agents disagree about information regarding a state that is irrelevant to a plan

under consideration.

The work of Toniolo et al. (2011) is closely related to our work, due to the focus on multiagent planning and the use of situation calculus. However, there are some important distinctions. In their work, argumentation is employed as a mechanism for the resolution of conflicts caused by the interdependencies between the plans of different agents, or the norms that individual agents have to abide by. On the contrary, our work focuses on contradictions in the agents specifications of the actions they can perform or the state of the environment. Dealing with contradictions is exactly the reason behind our choice to resort to a less expressive formalism than the language used in Toniolo et al. (2011).

#### 2.2.7.3 Defeasible Argumentation in Planning

Recent work in defeasible argumentation (Simari et al., 2004; García et al., 2007, 2008; Pardo et al., 2011) investigates the problem of planning when planning knowledge is combined with a set of additional defeasible rules. Simari et al. (2004) considers planning agents which are equipped, apart from their planning knowledge, with a set of defeasible rules that are applicable in every state of the world. States represent non-contradictory sets of facts. State transitions are revisions to the set of facts, which remove any literal that is complementary to an effect of the action, and then add the action's effects to this set. Action applicability is evaluated through a set of preconditions and constraints for each action. In order to account for defeasibility, action preconditions must be warranted from a knowledge base consisting of the set of facts forming the current state and the defeasible rules, whereas every constraint of the action must fail to be warranted.

The authors describe an algorithm for progression based planning, but mainly focus on a regression planning mechanism. The planner begins with a partial state containing the goal literals. A plan is built, with each action regressing to a previous state, until the initial state is reached. Extensions to this work (García et al., 2007, 2008) present DeLP-POP, an algorithm for regression-based partial-order planning. Dealing with partial states in a defeasible setting is particularly interesting. The addition of actions to the plan may cause the appearance of new defeaters, interfering with the existence of assumed warrants.

Pardo et al. (2011) extend DeLP-POP to accommodate cooperative multiagent scenarios. The problem setting involves multiple agents, sharing a common goal, but having different views of the planning domain, caused by knowledge about different

actions that may be performed, different views about the initial state and the defeasible rules governing the domain. The collection of all individual beliefs regarding the initial state is assumed to form a consistent set. The evaluation of plans, as well as search in the plan space is conducted as dialogue process. Plan evaluation corresponds to the collaborative search for threats to concrete plans. Search in the plan space is performed as $A^*$ search, guaranteeing completeness and optimality of the solution.

**2.2.7.3.1 Relevance to the Project** The multiagent extension to DeLP-POP is related to the problem of this thesis. However, there are some important distinctions. Contrary to Pardo et al. (2011) we do not assume that the collective initial state beliefs of the agents are consistent. In addition, we consider that agents may hold potentially contradicting specifications of the actions in the domains. Our planning language is more expressive, allowing variables and conditional effects.

#### 2.2.7.4 Argumentation in Multiagent Planning

The problem of multiagent planning in environments with non-deterministic actions and distributed, possibly inconsistent, information is the focus of recent work (Tang et al., 2009; Tang, 2012). Tang (2012) investigates the problem of coordinating the planing processes of multiple agents and coordinating joint plans. Planning in this approach is performed using symbolic model checking techniques. A defeasible factored action theory is introduced for the representation of multiagent state transitions. Two different types of specifications are employed to reason about the applicability of actions and the effectiveness of state transitions. This approach follows the Markov assumption that the computation of the successor state depends exclusively on the current state and the action causing the transition. Specifications are classified into layers representing frame information (FRM), operator effects (OP), agents local understanding of exceptions regarding effects (SLP) and agent interaction (IR). Inconsistencies among different specifications are resolved according to their respective layers and additional preference levels. It is assumed that IRs override SLPs, which override OPs, which in turn override FRMs. Centralised and decentralised planning algorithms are proposed to plan and coordinate the agents' joint behaviour. These algorithms are extended to allow planning with potentially inconsistent goals. The reasoning and planning procedures based on the defeasible factored action theory are re-formalised in an argumentation-theoretic fashion.

**2.2.7.4.1 Relevance to the Project** Research involving multiagent planning and argumentation is highly relevant to this thesis, and illustrates recent interest in the problem of planning with inconsistent theories. There are many similarities to our work, since both approaches deal with planning with contradictory information. However, there are also some important differences. Tang (2012) focuses on non-deterministic planning based on symbolic model checking. We maintain a close relation to classical planning, which allows the adaptation of efficient heuristic methods and the experimental evaluation of our approach using contradictory instances of benchmark planning problems.

Another major difference is related to the employed representation methods. The approach proposed by Tang (2012) is based on the logic of quantified boolean formulae and binary decision diagrams. On the contrary, our argumentation-based semantics are encoded on top of a defeasible situation calculus variant. We use defeasible successor state axioms to allow the compact representation of dynamic domains by encoding one axiom for every fluent literal. Moreover, our mechanisms can handle extended axioms that are not necessarily bound by the defeasible successor state axiom structure (which is useful to bind our formalism to the classical planning representation), as long as these can be encoded in the form of defeasible rules. Contrary to Tang (2012), because we consider preference information to be domain dependent, domain axioms are not explicitly prioritised depending on their type. Finally, in our work the evaluation of acceptability of claims regarding future situations is based on the acceptability of relevant information across the entire history leading to the final respective state.

#### 2.2.7.5 Argumentation-based Reasoning about Dynamic Domains

Previously in this chapter we discussed works on non-monotonic reasoning about action and dynamic domains. In a similar manner, argumentation-based reasoning has been used for enabling correct reasoning with concise and intuitive theories of action, while providing solutions to the frame (Kakas et al., 1999, 2001; Vo and Foo, 2005), qualification (Allen and Ferguson, 1994; Vo and Foo, 2001, 2005) and ramification problems (Kakas and Miller, 1997a; Vo and Foo, 2002, 2005). This section presents approaches in argumentation-based reasoning about action and discusses their relevance to this thesis.

Konolige (1988) argues in favour of the use of defeasible argumentation-based systems for reasoning about action, and describes which arguments are important for this type of reasoning and the information necessary to evaluate their status. This

approach is described based on the Yale Shooting Problem.

Ferguson and Allen (1994) consider the problem of plan communication in mixed initiative planning, and argue that a rich representation is necessary, allowing the communication of important goals and subgoals, relevant plans, clarifications and suggestions. Accordingly, they present an argumentation-based approach based on the *logic of time and action* (Allen and Ferguson, 1994). They propose a representation scheme based on defeasible rules and argue that it is better suited to deal with the qualification problem, enabling the agent to reason about qualifications for as much time as possible, while dealing with subsequently encountered qualifications given more time.

Kakas et al. (1999, 2001) focus on domains for reasoning about action written in the language $\mathcal{E}$ (Kakas and Miller, 1997b). They present a translation for these theories, enabling argumentation-based reasoning based on logic programming without negation as failure (Dimopoulos and Kakas, 1995). Reasoning in the language formalism $\mathcal{E}$ is based on *default persistence* to address the frame problem. The application of argumentation methods enables treatment of default persistence by prioritising effects of later actions higher than effects of earlier actions. In this approach time is considered to be totally ordered.

Vo and Foo (2001, 2002, 2005) present an argumentation-based framework designed to address the frame, qualification and ramification problems in a uniform manner. Argumentation-based reasoning follows assumption-based argumentation Bondarenko et al. (1997). Domain descriptions are based on a propositional action description language based on temporal logic (Sandewall, 1994; Drakengren and Bjäreland, 1999). Time is represented in terms of a sequence of discrete time points.

Augusto and Simari (2001) present an argumentation-based for temporal reasoning using the notions of instant and interval as temporal references. This system is based on temporal logic.

**2.2.7.5.1 Relevance to the Project** The focus of our work is different. We are interested in defeasibility introduced by the different opinions of the agents. Our representation is based on defeasible rules with the structure of successor state axioms. As as a result, each agent encodes both the frame and effect rules regarding a fluent predicate in a single axiom.

The implicit representation of the frame axioms allows reasoning about the different opinions on what does not change in the world after the application of the agents' actions. The different frame axioms held by different agents allow the implicit encod-

ing of the different views of not only what changes, but also what remains the same. Getting similar results from a defeasible representation would require the addition of rules, increasing the size of the theory. We revisit this in Chapter 6 where we compare our representation with an alternative defeasible representation based on default persistence.

Another benefit from the use of a formalism based on situation calculus is *branching time*. Branching time is useful for constructing arguments regarding different potential plans, especially when the agents engage in dialogue regarding different alternatives. The tree-like nature of branching time enables reuse of arguments and argument evaluation results for arguments extending a common initial plan.

## 2.3 Conclusions

Although there is an extensive body of work in the area of multiagent planning, there has been limited attention to the distributed and potentially erroneous nature of knowledge in multiagent systems. Most work dealing with the problems related to these aspects comes from the argumentation community, but does not deal with the problem of planning for multiple agents which have different perspectives about the environment. The related approaches mainly focus on the problem of deliberation rather than the planning-related aspects of means-end reasoning. In addition, these works do not deal with planning problems in the classical sense, and as a result either deal with simpler problems, such as action selection, or due to the complexity of the planning problem, lack the mechanisms for performing plan synthesis.

# Chapter 3

# Multi-Perspective Cooperative Planning

## 3.1 Introduction

The problem of multi-perspective cooperative planning (MPCP) arises when a coalition of autonomous agents, which hold incompatible views of the planning environment, need to come up with a plan that can be defended against possible objections. MPCP deals with the problem of planning with *distributed* and potentially *contradictory* knowledge, i.e. knowledge which contains directly contradicting facts or which leads to contradicting conclusions.

Contrary to classical planning, we treat agents' beliefs as evidence, rather than as certain facts. For instance, an agent's belief that a robot $r_1$ is at location $loc_1$ is treated as evidence suggesting that the robot is at $loc_1$, or that we have reasons to believe[1] that $r_1$ is at the suggested location. Accordingly, contradicting beliefs are viewed as evidence towards opposing conclusions. The agents can compare such evidence and identify which claims can be defended against possible objections.

The problem of multi-perspective cooperative planning consists of a *planning* and a *decision-making* problem:

1. Synthesise plans whose success is suggested by evidence from the agents' collective beliefs.

2. Evaluate the *acceptability* of these plans by comparing the evidence supporting

[1] We use the expressions indications, reasons to believe and evidence interchangeably. If such indications lead to contradicting conclusions, the beliefs used to reach these conclusions are called contradictory.

them against possible objections.

The first sub-problem is essentially a planning problem. We formalise it based on a standard set-theoretic planning representation, adapted to accommodate contradictions in the agents' beliefs and multiple operator specifications. This representation allows a close relation to classical planning enabling the use of efficient, off-the-shelf planners (McDermott, 2000; Hoffmann and Nebel, 2001) to solve the task at hand.

The planning sub-problem of MPCP focuses exclusively on the planning elements of the overall problem, stripped from any additional information the agents may hold regarding, for example, the source or credibility of planning beliefs. We specify a solution concept to this sub-problem and discuss how it qualifies as a solution to the overall problem.

The second problem involves deciding whether the agents should accept a synthesised plan. The formulation of this sub-problem requires a concrete account of the notion of acceptability. Based on argumentation theory (Dung, 1995; García and Simari, 2004) we formally specify these notions and concretise the problem of multi-perspective cooperative planning.

The proposed argumentation-based approach is based on a defeasible logic programming (García and Simari, 2004) and situation calculus (McCarthy, 1963) variant which enables the formulation of the problem and the methods in a purely logical, non ad-hoc, manner. This language is strictly more expressive than the set-theoretic notation used for the representation of the planning sub-problem. We provide a sound mechanism for the translation of set-theoretic planning theories to equivalent defeasible logic theories.

## 3.2 The Planning Problem of MPCP

This section focuses on the first sub-problem of MPCP. We focus on the agents' planning beliefs and disregard any additional information about the sources or credibility of these beliefs that can be used to provide additional insight on the evaluation of different alternatives.

We use the standard planning problem specification $P = \langle F, I, A, G \rangle$, with fluents $F$ initial state $I$ action space $A$ and goal state $G$, and consider the problem of working out a plan that complies with several (potentially contradictory) versions $P_i$ of $P$ at the same time, for each agent $i$. Agents maintain individual initial state beliefs $I_i$ and action specifications $A_i$. Fluents $F$ and goals $G$ are shared.

The notation used in this section is based on the STRIPS planning model (Fikes and Nilsson, 1972), extended with variables, negation immediately preceding fluent predicates and conditional effects. We consider all actions to be deterministic. At this point, we do not make any assumptions regarding the observability of the domain.

Variables are essential for the compact representation of a planning domain. The use of negation allows the representation of states as sets of literals (rather than atoms), and enables us to encode that a literal takes on a positive or a negative value, as well as absence of information regarding the value of a literal. Conditional effects enable a more concise specification of the operators. The resulting formal notation is sufficiently expressive for the compact representation of complex, deterministic planning theories.

We are not concerned with multiagent execution. Nevertheless, the employed planning model accommodates a simple execution model by including an additional term in each action specifying the agent executing the action. In order to account for joint actions, this scheme can be extended so that the agent term can be instantiated to a coalition of agents. The resulting model accounts for multiagent execution of fully ordered plans, and does not account for durative, concurrent actions.

### 3.2.1 Individual Planning Knowledge

Each individual agent's planning knowledge is encoded as an *individual planning problem*, defined on top of a logical language $\mathcal{L} = \langle \mathcal{L}_p, \mathcal{L}_v, \mathcal{L}_c \rangle$. $\mathcal{L}_p$ contains a finite number of predicate names. $\mathcal{L}_v$ is a finite set containing variable symbols, and $\mathcal{L}_c$ contains a finite number of constant symbols representing the objects of the planning domain. The fluents representing the domain are denoted by the tuple $F = \langle F_c, F_v \rangle$, where sets $F_c$ and $F_v$ contain ground and unground fluent literals respectively. $F_c$ is obtained by grounding the elements of $F_v$ using all possible objects from $\mathcal{L}_c$.

**Definition 10.** *The* individual planning problem *for agent i in a coalition of agents is a tuple:*

$$P_i = \langle F, I_i, O_i, G \rangle,$$

*where*

- $I_i \subseteq F_c$ *is agent i's perception of the* initial state *of the environment.*
- $O_i$ *is the set of* planning operators *i believes the coalition can apply.*

- $G \subseteq F_c$ *is a* goal *shared by the agents in the coalition.*

We assume that the goals shared by the agents are non-contradictory.

The set $O_i$ summarises agent $i$'s beliefs regarding the specification of the operators available to the agents in the coalition that can be used to reach the common goal. *Planning operators* are tuples of the form:

$$\langle pre_i, o, eff_i \rangle,$$

where

- $o$ is the name of the operator,
- $pre_i \subseteq F_v$ is the set of *preconditions* and
- $eff_i$ is a set of *conditional effects*.

We refer to beliefs of agent $i$ regarding the preconditions and effects of operator $o$ as $pre_i(o)$ and $eff_i(o)$ respectively.

Conditional effects have the form $\langle C, e \rangle$, where $C \subseteq F_v$ denotes the necessary conditions and $e \in F_v$ represents the effects. An action $a$ is a ground version of an operator, where each variable has been instantiated to a constant object. We denote the set of all ground actions for agent $i$ as $ground(O_i)$.

Contrary to classical planning, fluent literals can be considered to be indications, or reasons to believe that the literal takes on the respective value. Consider the following examples:

- $At(x, loc_1) \in I_i$ represents that agent $i$ has indications that object $x$ is at location $loc_1$ in the initial state.
- $pre_i(pickup(x, loc_1)) = \{At(x, loc_1), Movable(x)\}$ denotes that agent $i$ has reasons to believe that object $x$ can be picked up from location $loc_1$ if we have reasons to believe that $x$ is at $loc_1$ and that $x$ is transportable.
- $\langle \{Power(l)\}, Light(l) \rangle \in eff_i(switch\_on(l))$ represents agent $i$'s belief that, if there are indications suggesting that a lamp $l$ is connected to a power source, there are reasons to believe that $l$ is lit after the agent switches it on.

Planning domain beliefs can be used alongside the *state transition function* to calculate the effects that the application of an action is expected to have on the environment.

#### 3.2.1.1 State Transition Function

The *state transition function* specifies how operators alter the state of the world. It is usually represented as a function $\gamma(a,\sigma)$ which, given an action $a$ and the state of the environment $\sigma$, returns the state obtained after the execution of $a$.

Positive and negative literals may appear in the same state, representing different reasons suggesting that the respective atom takes a positive and a negative value in this state. We adapt the usual state transition function specification to handle literals instead of atoms in order to account for contradictory information.

The state transition function is defined for actions that the agent has reason to believe that they are applicable. Agent $i$ has reasons to believe that a ground action $a$ is *applicable* in a state $\sigma$ if there are reasons to believe that this state satisfies the preconditions of the action:

$$pre_i(a) \subseteq \sigma.$$

The following definition describes the specification of the state transition function in our setting. The proposed specification of the state transition function ensures that ambiguity about conditions in a state is propagated to literals in successor states whose value depends on these conditions.

**Definition 11.** *Consider agent i, and the individual planning problem* $P_i = \langle F, I_i, O_i, G\rangle$*, and a state* $\sigma_i$*. Let a ground action* $\langle pre_i, a, eff_i\rangle \in ground(O_i)$ *such that* $pre_i(a) \subseteq \sigma$*.* $e \in \gamma_i(a,\sigma)$ *if and only if:*

1. *there exists an effect* $\langle C, e\rangle \in eff_i(a)$ *and* $C \subseteq \sigma$*, or*
2. $e \in \sigma$ *and for every conditional effect* $\langle C, \overline{e}\rangle \in eff_i(a)$*, there exists* $c \in C$ *such that either*
   (a) $c \notin \sigma$*, or*
   (b) $\overline{c} \in \sigma$*.*

The notation $\overline{e}$ represents the complement of literal $e$. If $e$ is a positive literal then $\overline{e} = \neg e$, whereas if $e$ is a negative literal and $e = \neg p$, then $\overline{e} = p$.

Condition (1) describes that if there are reasons to believe that the conditions of a conditional effect hold, then there are reasons to believe that its effect holds in the successor state. This is similar to the usual specification of the state transition function.

Condition (2) deals with persistence of literals. The successor state function must account for the literals we have reasons to believe that they remain unaffected by the

application of the most recent action. In non-contradictory theories, calculating which literals remain the same simply involves deleting from the successor state the complements of the added literals. As a result, a literal remains unaffected if there is no conditional effect producing this literal whose conditions are all satisfied by the state.

Contradictions complicate this process. It is rational to conclude that, if the beliefs suggest that the conditions of the effect are not applicable, then there are no reasons to believe that the effect is valid. Accordingly, the successor state function must account both for absence of information, similar to the standard case, and existence of information to the contrary. Condition (2), accounts for the conclusion that the conditional effect is inapplicable, due to:

- Absence of information suggesting that the condition holds in the previous state (2a).
- Information that the condition does not hold in the previous state (2b).

Figure 3.1 illustrates the desirable behaviour of the state transition function. Consider an agent that is considering the effects of the action of flipping the light switch, while assuming that the light turns on if there is electricity in the building.

The transition from state $\sigma_1$ illustrates that if there are reasons to believe that there is power in the building, then there are reasons to believe that light will be on if the lamp is switched on. At the same time, since there is no ambiguity regarding whether there is power, there is no evidence to suggest that after the application of the action there is no light, and as a result $\neg light$ does not persist in $\gamma_i(switchOn, \sigma_1)$.

The transition from state $\sigma_2$ to $\gamma_i(switchOn, \sigma_2)$ describes that if there are no reasons to believe that the condition of the effect holds, then there is no indication towards the belief that the effect is applicable. As a result, the complement of the effect persists in the successor state. State $\sigma_3$ is similar to $\sigma_2$, but in this case there are reasons to believe that the condition of the effect actually does not hold in the predecessor state.

State $\sigma_4$ is ambiguous with respect to the predicate *power* as there are reasons to believe that both *power* and $\neg power$ are the case. The state transition propagates the ambiguity to the successor state $\gamma_i(switchOn, \sigma_4)$. Since *power* is satisfied in $\sigma_4$, there are reasons to believe that the conditional effect is applicable. As a result, there are reasons to believe that *light* holds in $\gamma_i(switchOn, \sigma_4)$. At the same time, there is evidence suggesting that $\neg power$ holds in $\sigma_4$. This leads to the conclusion that the conditional effect is not applicable, and accordingly, provides reasons to believe that

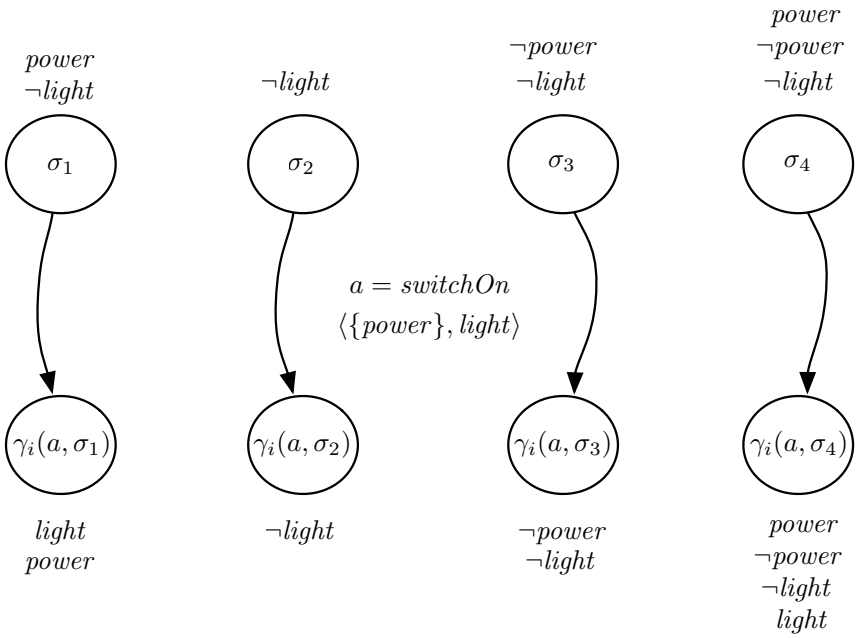



Figure 3.1: Example illustrating the specification of the state transition function for individual planning problems for different states $\sigma_1, \ldots, \sigma_4$ and a single action *switchOn* with the conditional effect $\langle\{power\}, light\rangle$. Fluent literals above and below states denote the beliefs that are part of the respective states

$\neg light$ holds in $\gamma_i(switchOn, \sigma_4)$. The literals *power* and $\neg power$ are not affected by the action *switchOn*. Their status persists to the successor state.

The state transition function propagates ambiguity to successor states, but does not introduce contradictions in non-contradictory states when the applicable effects of the action are not contradictory. The contradictions introduced are either a result of contradictions that persist from the previous state, or due to contradictory information regarding the conditions of conditional effects.

The state transition function for agent *i*, given an action *a* and the predecessor state *s* is represented using our set-theoretic notation as follows:

$$\gamma_i(a, \sigma) = (\sigma - del_i(a, \sigma)) \cup add_i(a, \sigma)\,.$$

The set $add_i(a, \sigma)$ contains the effects of the action *a* that agent *i* has reason to believe that are applicable:

$$add_i(a, \sigma) = \{e \mid \langle C, e\rangle \in eff_i(a) \text{ and } C \subseteq \sigma\}\,.$$

The set $del_i(a, \sigma)$ contains the literals that agent *i* has reason to believe no longer hold

after the application of the action:

$$del_i(a,\sigma) = \overline{add_i(a,\sigma) - inap_i(a,\sigma)}.$$

The set $inap_i(a,\sigma)$ contains the effects that are not believed to be applicable:[2]

$$inap_i(a,\sigma) = \{e \mid \text{ for every } \langle C,e\rangle \in eff_i(a), \exists c \in C \text{ such that } c \notin \sigma \text{ or } \overline{c} \in \sigma\}.$$

The bar notation over a set represents the complements of every literal in this set, i.e. $\overline{S} = \{\overline{e} \mid e \in S\}$.

The successor state is calculated as the union of the conditional effects that are believed to be applicable (i.e. $add_i(a,\sigma)$), and the literals which are believed to persist from the previous state (i.e. $\sigma - del_i(a,\sigma)$). In order to calculate the latter, we remove the conditional effects that are believed be applicable without any ambiguity from the predecessor state. This is asserted by removing from the applicable effects $add_i(a,\sigma)$, all effects which are believed to be inapplicable in $\sigma$ (i.e. $inap_i(a,\sigma)$).

The following proposition asserts that the set-theoretic specification of the state transition function follows Definition 11.

**Proposition 1.** *Consider agent i and the individual planning problem $P_i = \langle F, I_i, O_i, G\rangle$. Let $\langle pre_i, a, eff_i\rangle \in ground(O_i)$ be a ground action which is applicable in state $\sigma$. According to the specification of $\gamma$, $e \in \gamma_i(a,\sigma)$ if and only if:*

1. *there exists an effect $\langle C,e\rangle \in eff_i(a)$ and $C \subseteq \sigma$, or*
2. *$e \in \sigma$ and for every conditional effect $\langle C,\overline{e}\rangle \in eff_i(a)$, there exists $c \in C$ such that $c \notin \sigma$ or $\overline{c} \in \sigma$.*

*Proof.*
($\Rightarrow$)

1. If $\langle C,e\rangle \in eff_i(a)$ and $C \subseteq \sigma$ then $e \in add_i(a,\sigma)$, and as a result $e \in \gamma_i(a,\sigma)$.
2. We assume (a) that $e \in \sigma$ and (b) for every conditional effect $\langle C,\overline{e}\rangle \in add_i(a)$ there exists $c \in C$ such that $c \notin \sigma$ or $\overline{c} \in \sigma$. Therefore, $\overline{e} \in inap_i(a,\sigma)$, and as a result $e \notin del_i(a,\sigma)$. Therefore, $e \in \gamma_i(a,\sigma)$.

($\Leftarrow$) Assume that

[2] More specifically, this set contains the effects that the agent has no reason to believe that they are applicable, or has reasons to believe that they are inapplicable.

1. there does not exist an effect $\langle C, e\rangle \in add_i(a)$ with $C \subseteq \sigma$, and

2. (a) $e \notin \sigma$ or (b) there exists a conditional effect $\langle C, \overline{e}\rangle \in eff_i(a)$, such that for all $c \in C$, $c \in \sigma$ and $\overline{c} \notin \sigma$.

From (1) it follows that $e \notin add_i(a,\sigma)$. Then, if $e \notin \sigma$, it holds that $e \notin \gamma_i(a,\sigma)$. If (2b) is the case, $\overline{e} \in add_i(a,\sigma)$, and also $\overline{e} \notin inap_i(a,\sigma)$. As a result $e \in del_i(a,\sigma)$. Therefore, regardless of whether $e \in \sigma$ holds, it is the case that $e \notin \gamma_i(a,\sigma)$. □

The state transition function does not introduce contradiction in a non-contradictory state, when the action causing the transition does not have any contradicting effects.

**Proposition 2.** *Consider agent i and the individual planning problem $P_i = \langle F, I_i, O_i, G\rangle$. Let $\langle pre_i, a, eff_i\rangle \in ground(O_i)$ be a ground action which is applicable in state $\sigma$. If $\sigma$ is not contradictory (i.e. $\nexists p \in \sigma$ such that $\overline{p} \in \sigma$) and all applicable effects (i.e. $add_i(a,\sigma)$) are non-contradictory, then $\gamma_i(a,\sigma)$ is non-contradictory, for any action a.*

*Proof.* For every $e \in add_i(a,\sigma)$, there exists a conditional effect $\langle C, e\rangle \in eff_i(a)$ such that $C \subseteq \sigma$. Since $\sigma$ is non-contradictory, there does not exist any $c \in C$ such that $\overline{c} \in \sigma$. As a result, $inap_i(a,\sigma)$ is empty and as a result it holds that:

$$del_i(a,\sigma) = \overline{add_i(a,\sigma) - inap_i(a,\sigma)} = \overline{add_i(a,\sigma)}\,.$$

Therefore, the state transition is calculated is follows:

$$\gamma_i(a,\sigma) = (\sigma - del_i(a,\sigma)) \cup add_i(a,\sigma) = (\sigma - \overline{add_i(a,\sigma)}) \cup add_i(a,\sigma)\,.$$

Accordingly, for every literal added to the predecessor state, its complement is removed. As a result $\gamma_i(a,\sigma)$ is non-contradictory. □

#### 3.2.1.2 Candidate Plans

Desirable states may not be reachable by performing single actions. *Candidate plans* are sequences of actions, which are believed to be applicable in sequence, and reach a desirable state.

**Definition 12.** *A sequence of actions $\pi = \langle a_1, \ldots, a_k\rangle$ is a* candidate plan *for $P_i$ if*

- *$a_1$ is applicable in I,*

- *for every action $a_j$, with $2 \leq j \leq k$, $a_j$ is applicable in state $\gamma_i(\langle a_1, \ldots, a_{j-1}\rangle, I)$ following the application of $a_1, \ldots, a_{j-1}$, and*

- $G \subseteq \gamma_i(\pi, I)$.

The first condition accounts for the executability of the plan, meaning that all actions in the plan are applicable in sequence. The second condition asserts that $i$ has reasons to believe that the sequence achieves the goal. In order to accommodate the treatment of plans, the state transition function $\gamma_i$ is canonically extended to sequences of actions.

### 3.2.2 Collective Planning Knowledge

The multi-perspective planning problem is formulated with respect to the agents' collective planning beliefs.

**Definition 13.** *A* multi-perspective planning problem *is a tuple* $P = \langle N, F, I, O, G \rangle$, *where*

- $N$ *is the set of the agents participating in a coalition,*
- $F$ *are the fluents describing the domain,*
- $I = \{I_i\}_{i=1}^n$ *is a set containing all agents' views of the initial state,*
- $O = \{O_i\}_{i=1}^n$ *contains every agent's perception of the operators that the coalition may apply, and*
- $G \subseteq F_c$ *is a non-contradictory set describing the common goal.*

This definition is based on the assumption that the agents are cooperative, since they share the same goal $G$. In addition, we assume that the agents operate under ontological agreement, since they share the same fluents and operator names. On the contrary, we do not make any assumptions that initial states are individually and mutually non-contradictory, or that they maintain the same operator specifications.

The state transition function for the multiagent planning problem is defined as

$$\gamma(a, \sigma) = \{l \mid l \in \gamma_i(a, \sigma) \text{ for every } i \in N \text{ such that } pre_i(a) \subseteq \sigma\} \cup \{l \mid l \in \sigma \text{ for every } i \in N \text{ such that } pre_i(a) \not\subseteq \sigma\} .$$

The collective state transition function aggregates the results of individual state transition functions. If an action is not applicable with respect to an agent's operator specification, then the state transition function for this agent is undefined. In this case, the action is considered to have no effect on the state with respect to the specification of

this agent. Clearly, the resulting states may be contradictory either due to contradictory operator specifications, or contradictory initial state beliefs.

Equivalently to the individual agent's case, a sequence of actions is a candidate solution to the collective MPCP problem, if there are collective reasons to believe that the plan is executable and the goal is achieved in the resulting state.

**Definition 14.** *A sequence of actions* $\pi = \langle a_1, \ldots, a_m \rangle$ *is a* candidate plan *for a MPCP problem P if:*

- *there exists a tuple* $\langle pre_i, a_1, post_i \rangle \in ground(O)$ *such that* $pre_i(a_1) \subseteq I$,
- *for every action* $a_j$, *with* $2 \leq j \leq m$, *there exists* $\langle pre_k, a_j, post_k \rangle \in ground(O)$ *such that* $pre_k(a_j) \subseteq \gamma(\langle a_1, \ldots, a_{j-1} \rangle, I)$, *and*
- $G \subseteq \gamma(\pi, I)$.

The definition asserts that there is at least one specification of $a_1$ that is applicable in $I$, and for every action $a_j$ in the plan there is at least one specification that is applicable in the resulting state reached after executing the actions in the plan before $a_j$. The second condition ensures that the agents have reasons to believe that the goal is satisfied in the state resulting from the application of the plan.

Candidate plans that solve a collective planning problem include the candidate solutions to the individual agents' planning problems.

**Proposition 3.** *Let* multi-perspective planning problem $P = \langle N, F, I, O, G \rangle$, *and an individual planning problem* $P_i = \langle F, I_i, O_i, G \rangle$. *If* $\pi = \langle a_1, a_2, \ldots \rangle$ *is a candidate plan for* $P_i$, *then* $\pi$ *is a candidate plan for P.*

*Proof.* Every action $a$ that is applicable in $\sigma$ with respect to $P_i$ is also applicable in $\sigma$ with respect to $P$, since $\langle pre_i, a, eff_i \rangle \in O_i \Rightarrow \langle pre_i, a, eff_i \rangle \in O$, and $pre_i \subseteq \sigma$ as $a$ is applicable in $\sigma$ with respect to $P_i$. Therefore, since $I_i \subseteq I$ and $pre_i \subseteq I_i$, $pre_i \subseteq I$.

From the specification of $\gamma$ we infer that, for every action $a$ that is applicable in a state $\sigma$, $\gamma_i \subseteq \gamma$. As a result for every action $a_j$ in the plan with $j > 1$, it holds that $pre_i \subseteq \gamma(\langle a_1, \ldots, a_{j-1} \rangle, I)$, since $pre_i \subseteq \gamma_i(\langle a_1, \ldots, a_{j-1} \rangle, I)$. In addition, $G \subseteq \gamma(\langle a_1, \ldots, a_{j-1} \rangle, I)$, since $G \subseteq \gamma_i(\langle a_1, \ldots, a_{j-1} \rangle, I)$.

Therefore, every action in $\pi$ is applicable in sequence with respect to $P$ and the final state satisfies the goal conditions $G$. □

The specification of candidate plans asserts that reasons for executability and goal achievement derive from the agents' collective beliefs. However, they do not take into

account possible objections. These bring about doubts regarding the a plan based on indications that certain actions are not executable or that the plan fails to achieve the goal.

Consider, for instance, a candidate plan, the first action of which is believed to be executable because there are reasons to believe that its precondition $p$ is satisfied in the initial state. However, if at least one agent in the coalition has reasons to believe that $\neg p$ is the case, the agents have collective reasons to believe that the first action in the plan is not applicable. In such cases, it is not straightforward to decide whether the plan should be followed. Candidate plans can be viewed as weak solution concepts as they indicate possible solutions, which derive from the agents' collective beliefs, but are not objection-proof.

The next section focuses on the notion of acceptability from argumentation theory, and provides the formal semantics for the resolution of contradictions and the evaluation of the evidence supporting candidate plans. Planning knowledge models the bare minimum knowledge that is required to synthesise plans, and cannot represent additional meta-knowledge about these beliefs, such as useful information regarding levels of confidence, origin or structure. In the next section we describe how such beliefs can be utilised in a unified framework, enabling the agents to resolve domain knowledge contradictions and make sound decisions.

## 3.3 Defeasible Reasoning about MPCP Problems

In the previous section we outlined the basic components of the planning sub-problem of MPCP. Solving this problem, in the traditional planning sense, results in synthesising candidate plans. These plans, however, are greedy as the planner may use any beliefs even if contradicting views about these beliefs exist.

This section introduces the notion of *acceptability*, which provides the basis of stronger solution concepts. Plan acceptability is based on the idea that plans must not only derive from the agents' collective beliefs, but these beliefs must be "stronger" than any objections.

We follow an argumentation-based approach, where derivations from the agents' collective beliefs correspond to arguments. Arguments are compared in order to identify which conclusions the agents should accept. According to Prakken and Vreeswijk (2002), the specification of an argumentation system requires the definition of the following elements:

- An underlying logical language.
- Definitions of arguments, conflicts between arguments and of a defeat relation among arguments.
- Definition of how arguments are assessed, specifying a notion of defeasible logical consequence.

This section provides a concrete specification of these notions for the problem of reasoning about the acceptability of plans.

The set-theoretic formalism introduced in the previous section provides a formal specification of the planning problem. This reasoning scheme allows agents to use their (collective) planning beliefs to evaluate whether they have reasons to believe that a literal holds after the application of a sequence of actions. However, even though this formalism allows the logical specification of individual states, it is inadequate for the representation of logical statements explaining how derived conclusions regarding different states are related.

In order to resolve contradictions regarding in future states, agents need to inspect all relevant beliefs that lead to these contradictions. This requires regressing to previous states and identifying relevant beliefs and related meta-knowledge the agents may possess. In order to resolve conflicts, the agents must be able to find all the explanations relevant to a contradiction, and decide which one is the 'strongest'.

We map derivations to arguments, and employ abstract argumentation techniques to formally specify "acceptable plans". The set-theoretic, state-based planning representation does not allow us to formulate logical arguments based on some form of deductive reasoning. In order to avoid imposing ad-hoc internal structure and semantics, we introduce a defeasible variant of the situation calculus language that provides the basis for a structured logical representation of arguments which in turn will be used as the foundation of our plan acceptability semantics.

### 3.3.1 Desirable Properties of the Logical Formalism

Before getting into the details of our logical formalism, we outline a series of essential properties. In Chapter 6 we revisit these properties in order to evaluate the suitability of the employed formalism.

#### 3.3.1.1 Representation of MPCP Domains

The formalism must be sufficiently expressive to provide an accurate representation of multi-perspective planning problems. Also, the inferential results provided by the formalism should be correct with respect to the aforementioned solution concepts.

#### 3.3.1.2 Reasoning about Dynamic domains

The employed logical formalism must be able to represent domains and change caused by the actions of the agents. It must allow axioms describing the effects and preconditions of actions. Additionally, it is necessary to be able to formulate sentences describing the state of the environment, and how the values of literals change throughout the execution of the plan.

#### 3.3.1.3 Expressive Power

The expressive power of the formalism is also relevant to the types of axioms that can be represented. We have the basic requirement that the formalism allows the expression of axioms describing planning operators. More expressive formalisms may enable the more elegant representation of theories including more rules explicitly describing concepts such as domain constraints and ramifications.

#### 3.3.1.4 Handing Contradictions

The formalism must enable reasoning with contradictory theories. The inferential mechanism must be able to handle contradictions and provide concrete, intuitive semantics to specify what is acceptable. Since there is no universally accepted measure of what constitutes adequate justification to accept new information, a single acceptability criterion may not be adequate. Multiple semantics may be necessary to account for different agent attitudes towards accepting new facts. The combination of the requirements of reasoning about dynamic domains and handling inconsistencies are the focal points of our approach.

#### 3.3.1.5 Practicality

Both the aspects of tractability of the inferential mechanism and the size of theory determine the practicality of the approach. Expressive formalisms produce succinct representations. However, complicated rules generally increase the complexity of the

reasoning process. The practicality of the approach is related to this tradeoff, and can be measured by its ability to handle domains of increasing size, complexity and number of contradictions.

### 3.3.2 Defeasible Situation Calculus

Situation calculus (McCarthy, 1963) is a highly expressive logical language for reasoning about dynamic domains. Highly structured theories of situation calculus, called *basic action theories*, enable tractable reasoning, while providing a solution to the frame problem (Reiter, 1991, 2001). Basic action theories, however, do not normally cater for contradictory planning knowledge, which complicates the reasoning process.

The collective beliefs of multiple agents are may be contradictory and may include multiple specifications for the same operator. Also, the task of combining distinct specifications held by multiple agents to formulate well-formed successor state axioms, pre-compiling the collective beliefs into a non-contradictory well formed basic action theory, is not a straightforward task.

In order to be able to handle the additional complexity introduced by contradictions and multiple operator specifications, we focus on a restricted variant of situation calculus based on defeasible logic programming, which we call *defeasible situation calculus*. Defeasible logic programming enables defeasible, argumentation-based reasoning with propositional theories that may contain contradictory beliefs. The representation resembles the representation of extended logic programming (García and Simari, 2004), and is syntactically similar to basic action theories that have transformed to be interpreted by the situation calculus Prolog interpreter (Reiter, 2001).

#### 3.3.2.1 The language $\mathcal{L}_{defsitcal}$

The language of defeasible situation calculus $\mathcal{L}_{defsitcal}$ supports three basic sorts:

- *action* for actions
- *situation* for situations
- *object* for everything else

$\mathcal{L}_{defsitcal}$ has the following alphabet:

- Usual DeLP negation '$\sim$', default negation '*not*', conjunction ',' and defeasible implication '$-\!\prec$' connectives.

- Equality '=', inequality '$\neq$' and disjunction ';' symbols.
- Countably infinitely many individual variable symbols of each sort.
- Two function symbols of sort situation:
  1. A constant symbol $S_0$, denoting the initial situation.
  2. The binary function symbol $do : action \times situation \rightarrow situation$.
- A binary predicate symbol $Poss : action \times situation$.
- A finite or countably infinite set of predicate symbols with arity $n$, for each $n \geq 0$, and sorts $(action \cup object)^n$.
- A finite or countably infinite number of predicate symbols of sort $(action \cup object)^n \times situation$, for each $n \geq 0$.

Situations are interpreted as finite sequences of actions. The special function symbol $do(a,s)$ represents the sequence formed by adding an action $a$ to the sequence $s$. For readability we write $do([a_1,\ldots,a_n],s)$ to denote $do(a_n,do(\ldots do(a_1,s))],s)$. Situations that are subsequences of other situations are referred to as predecessors to these situations. For instance, $s_2 = do([a_1,a_2],s)$ is a predecessor of $s_3 = do([a_1,a_2,a_3,a_4],s)$. We call $s_3$ a successor situation of $s_2$.

The special symbol $Poss(a,s)$ denotes that action $a$ can be applied in situation $s$. Predicates are distinguished from fluents and non-fluents. Fluent predicates have an object of term situation as their final argument. Non-fluents describe situation-independent relations that do not change over time. Note that situation terms are allowed to appear exclusively as the last argument of fluent predicates, or special symbols *do* and *Poss*.

Default negation is interpreted, similar to García and Simari (2004), as an assumption about the absence of contradicting information. On the contrary, the usual negation denotes reasons to believe that a literal takes a negative value.

Equality, inequality and disjunction symbols do not exist in DeLP. We introduce these symbols to relate terms, as for instance $a_1 = a_2$ and $x_1 \neq x_2$, for actions $a_1$, $a_2$ and objects $x_1$, $x_2$. Equality and inequality are treated, with respect to reasoning, as grounding constraints, rather than logical symbols. We also allow the use of disjunction to enable the compact representation of multiple rules with the same body using a single rule. The effect of these symbols in the reasoning process is further explained after the introduction of our reasoning mechanism.

We use $\mathcal{L}_{defsitcal}$ to encode *defeasible situation axioms*, which have the following form:

$$head \prec\!\!\!- body\,.$$

The head of the rule may be any literal constructed for any predicate symbol in $\mathcal{L}_{defsitcal}$ or the special predicate symbol $Poss(a,s)$, where $a$ is an action and $s$ a situation term. The body of the rule may be empty. Alternatively, it may contain any literal for any predicate symbol in $\mathcal{L}_{defsitcal}$, or equalities and inequalities for any terms in $\mathcal{L}_{defsitcal}$. Literals in the body of axioms may be preceded by default negation. Conjunction may appear between literals, default negated literals, term equalities and inequalities. Disjunction is only allowed to separate multiple conjunctions. The body of axioms is written in a form of *disjunctive normal form* adapted to account for default negation.

This work has been designed with practicality in mind. The restrictions in the structure of axioms serve this purpose. By following these rules we allow the axiomatisation of planning domains in an ungrounded manner, while enabling a specification of an efficient grounding scheme that generates propositional theories that can enable efficient reasoning. More specifically, while we allow the use of disjunction to enable the specification of complex axioms, we restrict its use so that it enables us to quickly generate propositional theories that can be used for DeLP-style inference. Regardless of these restrictions, the proposed axiomatisation scheme allows the specification of complex domains. We revisit this issue in Section 3.4.1.3.

The formal grammar is outlined in Backus-Naur Form as follows:

1. $\langle axiom\rangle ::= \langle head\rangle \prec\!\!\!- \langle body\rangle \mid \langle action\rangle = \langle action\rangle \mid \langle action\rangle \neq \langle action\rangle$
2. $\langle head\rangle ::= \langle literal\rangle \mid [\sim]\langle poss\rangle$
3. $\langle body\rangle ::= [\langle disjunct\rangle]$
4. $\langle disjunct\rangle ::= \langle conjunct\rangle[;\langle disjunct\rangle]$
5. $\langle conjunct\rangle ::= \langle element\rangle[,\langle conjunct\rangle]$
6. $\langle element\rangle ::= [not]\langle literal\rangle \mid \langle grounding\text{-}constraint\rangle$
7. $\langle literal\rangle := [\sim]\langle predicate\rangle$
8. $\langle grounding\text{-}constraint\rangle ::= \langle term\rangle = \langle term\rangle \mid \langle term\rangle \neq \langle term\rangle$

The expression $\langle poss \rangle$ refers to the special predicate $Poss(a,s)$.

Following standard DeLP, axioms with an empty body are called presumptions (García and Simari, 2004). Sometimes instead of $head \prec$ we write $head$ for simplicity.

According to the aforementioned restrictions the following axiom is not well-formed:

$$\sim Lit(l, do(a,s)) \prec \sim Lit(l,s), not((a = switch\_on, Plugged(l,s)); \\ (a = switch\_on, \sim Broken(l,b,s)).$$

On the contrary, the next rule illustrates an example of a well-formed axiom:

$$\sim Lit(l, do(a,s)) \prec \sim Lit(l,s), a \neq switch\_on, a \neq switch\_on; \\ \sim Lit(l,s), a \neq switch\_on, not \sim Broken(l,b,s); \\ \sim Lit(l,s), not\, Plugged(l,s), a \neq switch\_on; \\ \sim Lit(l,s), not\, Plugged(l,s), not \sim Broken(l,b,s).$$

#### 3.3.2.2 Notational Conventions

Universal quantifications are not explicitly stated following the usual situation calculus convention. Lower-cased symbols denote unground variables quantified with maximum scope, unless stated otherwise. Constants are represented using capital letters. The representation $\vec{x}$ denotes $x_1, x_2, \ldots, x_n$. The notation $L(\vec{x})$ describes that the free variables in $L$ are among $\vec{x}$.

We follow the usual situation calculus naming conventions:

- Predicate names start with an upper case first letter.
- Function names are represented with a lower case first letter.
- Unground variables are denoted as lower case letters.
- Grounded variables are represented as capitalised letters.

In addition to the (defeasible) situation calculus representation, we need to also represent arguments and dialogue moves. To avoid confusion we represent arguments using lower case Greek characters.

### 3.3.3 Defeasible Basic Action Theories

The form of defeasible situation calculus axioms we described earlier is very liberal. Here we impose a more restrictive structure that is sufficient to represent planning domains. We call these theories *defeasible basic action theory* (DBAT). DBATs are influenced by Reiter's situation calculus basic action theories, and use defeasible successor-state axioms, which provide a solution to the frame problem by encoding both the frame and effect rules regarding a fluent predicate within an axiom. The result of this structure is a succinct representation of the planning domain.

**Definition 15.** *A* defeasible basic action theory *is a tuple* $\mathcal{D} = \langle \mathcal{D}_{ss}, \mathcal{D}_{ap}, \mathcal{D}_{S_0}, \mathcal{D}_{una}, \mathcal{D}_c \rangle$ *that contains defeasible rules describing successor state axioms* $\mathcal{D}_{ss}$*, action preconditions axioms* $\mathcal{D}_{ap}$*, axioms regarding the initial situation and non-fluent beliefs* $\mathcal{D}_{S_0}$*, the unique names axioms for actions* $\mathcal{D}_{una}$ *and constant symbols of sort object* $\mathcal{D}_c$*.*

*Defeasible successor-state axioms* detail conditions, providing reasons to believe that a fluent literal holds in the successor situation. The head of a defeasible successor state axiom is a fluent literal in a successor situation $do(a,s)$ (e.g. $F(do(\vec{x},a,s))$ or $\sim F(do(\vec{x},a,s))$), and the body of the rule involves only fluent predicates referring exclusively to situation term $s$. Examples of successor state axioms are the following:

$$Lit(do(a,s)) \prec a = switch\_on; Lit(s), a \neq switch\_off$$
$$Charged(x, do(a,s)) \prec Plugged(x,s), a \neq unplug(x)$$
$$\sim Charged(x, do(unplug(x),s)) \prec$$

Each agent holds one successor state axiom per fluent literal. This axiom must account for any possible case that leads to the indication that the literal holds in the successor state.

*Defeasible Action precondition axioms* denote reasons that govern the applicability of an action. The head of these rules is the predicate $Poss(A,s)$ or $\sim Poss(A,s)$, where $A$ is a ground action. Fluents in the bodies of these rules are not allowed to contain any situation terms apart from $s$. The following are examples of defeasible action precondition axioms:

$$Poss(switch\_on(l),s) \prec Reachable(l,s), \sim Broken(l,s)$$
$$\sim Poss(unplug(l),s) \prec \sim Reachable(l,s)$$

*Axioms regarding the initial situation* are rules which only contain predicates that have the initial situation as their situation term. For example, consider the following:

$$At(R, L_1, S_0) \prec$$
$$\sim At(R, L_1, S_0) \prec At(R, L_2, S_0), L_2 \neq L_1$$

Non-fluent beliefs never change from one situation to another (i.e. as a result of action execution). Objects of sort situation are not allowed to appear in such beliefs. Examples of non-fluent beliefs are the following:

$$Robot(R) \prec$$
$$Mobile(R) \prec Robot(R)$$

Regardless of their constant nature, and similar to every other belief in the theory, non-fluent beliefs are defeasible and may contradict each other.

Equality and inequality are just used to disregard ground instances of defeasible rules that do not respect these. For instance, given a successor state axiom that has the equality $a = A_1$, instantiating $a$ with $A_2$ is not possible since $A_1$ and $A_2$ are different objects. We use the set of unique names axioms for our domain to encode these constraints.

### 3.3.4 Grounding Defeasible Basic Action Theories

In order to utilise the reasoning mechanism of García and Simari (2004) we introduce a grounding mechanism for defeasible basic action theories. This mechanism produces well-formed extended defeasible logic programs.

A ground defeasible basic action theory is obtained after grounding the defeasible rules in the domain with respect to a ground situation term and all its predecessor situations. All ground situation terms are rooted in the initial situation $S_0$, and denote the history of the application of sequence of ground actions in the initial situation $S_0$. Note that grounding the domain theory for an extensive number of situation terms is an expensive process. We revisit this issue in the following chapter and present algorithms for reasoning and planning with DBATs.

**Definition 16.** *Let a defeasible basic action theory* $\mathcal{D} = \langle \mathcal{D}_{ss}, \mathcal{D}_{ap}, \mathcal{D}_{S_0}, \mathcal{D}_{una}, \mathcal{D}_c \rangle$. $ground(\mathcal{D}, S)$ *represent the DBAT* $\mathcal{D}$ *grounded with respect to a ground situation symbol S and be specified as follows:*

1. *Collect all ground rules from $\mathcal{D}$ for every possible grounding using all terms of sort object, every grounded action and every ground situation term in the set $\{S\} \cup \{S' \mid S'$ is a predecessor to $S\}$:*

$$\begin{aligned} &ground\_axioms(\mathcal{D}, S) = \\ &\quad \{r' \mid r \in \mathcal{D}_{ss} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{S_0} \text{ and } r' \text{ is obtained by the substitution} \\ &\qquad \text{of every unground object, action and situation term} \\ &\qquad \text{appearing in the rule with a ground term of the same sort} \\ &\qquad \text{and the removal of trivial equalities of the form } X = X \text{ and} \\ &\qquad\quad \text{inequalities } X \neq Y \text{ for different ground terms } X \text{ and } Y\}. \end{aligned}$$

2. *Simplify the grounded rules by separating disjunctions in their body into separate rules:*

$$\begin{aligned} &simplified\_ground\_axioms(\mathcal{D}, S) = \\ &\quad \{r \mid r \in ground\_axioms(\mathcal{D}, S) \text{ and the disjunction symbol} \\ &\qquad \text{does not appear in } r\} \cup \\ &\quad \{L \prec\!\!\!- \Phi \mid L \prec\!\!\!- \Psi \in ground\_axioms(\mathcal{D}, S) \text{ and } \Phi \text{ is a disjunct in } \Psi\}. \end{aligned}$$

3. *Remove rules which contain equalities referring to different objects and inequalities referring to the same object in their bodies:*

$$\begin{aligned} &ground(\mathcal{D}, S) = \{r \mid r \in simplified\_ground\_axioms(\mathcal{D}, S) \\ &\qquad \text{and for any different terms } X, Y, \text{ the statement} \\ &\qquad\quad X = Y \text{ or } X \neq X \text{ does not appear in the body of } r\}. \end{aligned}$$

$ground(\mathcal{D}_{ss}, S)$ and $ground(\mathcal{D}_{ap}, S)$ represent the ground defeasible successor state axioms and action preconditions respectively. The initial situation axioms do not contain variables and therefore remain unchanged. The ground defeasible basic action theory can be represented as the tuple:

$$ground(\mathcal{D}, S) = \langle ground(\mathcal{D}_{ss}, S), ground(\mathcal{D}_{ap}, S), \mathcal{D}_{S_0} \rangle.$$

In order to obtain the ground versions of the rules we instantiate their variables in all possible combinations and remove trivial equalities ($X = X$) and inequalities

($X \neq Y$). Then, we simplify the axioms by breaking down disjunctions in their bodies. Due to the specific structure we impose on the axioms their body is written in disjunctive normal form (slightly adapted to account for default negation), making the simplifications of the axioms a trivial process.

After the grounding process, ground action and situation terms are treated as objects with respect to the reasoning process. These objects are equal if and only if they have the same symbol name, arity and arguments. For simplicity, occurrences of the special function symbol *do* and actions $A(\vec{X})$ may be substituted with an equivalent unique ground situation and action terms respectively, in a uniform manner across the theory. For instance, the action $move(L1, L2)$ may be substituted with the new term $A_1$. Equivalently, $do(A_1, S_0)$ may be substituted with $S_1$ and $do([A_1, A_2], S_0)$ may be replaced by $S_2$.

Every axiom in $ground(\mathcal{D}_{ss}, S)$ is a well-formed *extended defeasible rule*, since heads of the rules are ground literals, and bodies of the rules are conjunctions of ground literals (possibly preceded by default negation). Therefore, every ground DBAT corresponds to an *extended defeasible logic program*. These programs do not contain strict rules. Based on this relation, we follow the reasoning mechanism of DeLP (García and Simari, 2004).

### 3.3.5 Defeasible Derivations

Defeasible rules are used in sequence to create inference chains called *defeasible derivations*. A defeasible derivation provides evidence for the derived conclusions. The following definition is slightly adapted from García and Simari (2004) to account for lack of strict rules in ground DBATs.

**Definition 17.** *Let $ground(\mathcal{D}, S)$ be a set of grounded defeasible rules and L a ground literal. $ground(\mathcal{D}, S) |\sim L$ represents a* defeasible derivation *of L from $ground(\mathcal{D}, S)$, which consists of a finite sequence $L_1, \ldots, L_n = L$ of grounded literals. For each literal $L'$ in the sequence, there exists a rule $r \in ground(\mathcal{D}, S)$ with $head(r) = L'$, and all literals appearing in its body, except the ones preceded by default negation, appear in the sequence, before $L'$.*

Note that in the special case that $L$ is a non-fluent symbol, grounding is performed only with respect to the initial situation $S_0$.

Defeasible derivations are monotonic. Introducing additional rules in a theory (possibly) expands the set of defeasible derivations that can be made using this theory.

**Proposition 4.** *Assume a defeasible basic action theory $\mathcal{D}$, and two ground situation terms $S_k$ and $S_l$ such that $S_k$ is predecessor of $S_l$. For every ground situation term $S_i$ and every ground fluent literal L, if ground$(\mathcal{D},S_k)\,|\!\sim L(S_i)$ then ground$(\mathcal{D},S_l)\,|\!\sim L(S_i)$.*

*Proof.* Definition 16 states that ground defeasible basic action theories include all axioms grounded for all combinations of ground terms that respect the expressed equalities and inequalities. The ground theories $ground(\mathcal{D},S_l)$ and $ground(\mathcal{D},S_k)$ have been grounded with exactly the same constants, apart from the additional situation terms that are successor to $S_k$ and predecessor or equal to $S_l$. Therefore, $ground(\mathcal{D},S_l) \subset ground(\mathcal{D},S_k)$, and as a result if $ground(\mathcal{D},S_l)\,|\!\sim L(S_i)$ then $ground(\mathcal{D},S_k)\,|\!\sim L(S_i)$. □

This proposition asserts that any derivations relevant to a plan $\pi$ are also relevant to every plan that begins with the sequence $\pi$. As a result, derivation results regarding $\pi$ may be reused when focusing on any plan extending $\pi$.

For readability purposes, we write $\mathcal{D}\,|\!\sim L(S)$ to represent that there is a defeasible derivation $ground(\mathcal{D},S)\,|\!\sim L(S)$. As shown by the previous proposition, the overloading of the defeasible derivation notation does not lead to erroneous conclusions due to the monotonicity of the defeasible derivation relation.

### 3.3.6 Acceptability

Defeasible derivations identify conclusions that can be derived from the agents beliefs. However, they do not investigate the tenability of these claims. This section introduces the notion of arguments and provides a concrete specification of the criteria that qualify acceptable reasoning about plans. The following analysis is based on García and Simari (2004).

#### 3.3.6.1 Arguments

*Arguments* capture reasons supporting a claim. They are minimal, non-contradictory sets of rules that defeasibly entail a conclusion. A set of rules is non-contradictory if there exists no literal which can be defeasibly inferred from the set, and its complement can also be inferred from the same set.

**Definition 18.** *Let ground$(\mathcal{D},S)$ be a set of ground defeasible rules and h a ground literal. $\alpha = \langle \mathcal{B},h \rangle$ is an* argument *for h if $\mathcal{B} \subseteq ground(\mathcal{D},S)$ and:*

1. $\mathcal{B} \mid\sim h$,
2. $\mathcal{B}$ *is non-contradictory, and*
3. $\mathcal{B}$ *is minimal, meaning that there does not exist a proper subset of* $\mathcal{B}$ *that satisfies conditions (1), and (2).*

The functions $Claim(\alpha) = h$ and $Support(\alpha) = \mathcal{A}$ denote the claim and support of the argument $\alpha = \langle \mathcal{A}, h \rangle$ respectively.

Arguments are constructed from ground theories in which axioms containing disjunctions have been simplified. As a result, the support sets contain rules whose bodies are conjunctions of literals. Therefore, we only include the relevant disjuncts of the original compound rule. So if we construct an argument using $F \prec\!\!\!- L_2$, from the original rule $F \prec\!\!\!- L_1; L_2; L_3$, we include only $F \prec\!\!\!- L_2$ in the support set of this argument. The rational behind this practice is related to the minimality constraint that is imposed on the support of arguments. By excluding unnecessary information, we reduce the support size and simplify the evaluation of the acceptability of arguments as we minimise the supporting beliefs that need to be investigated, excluding all irrelevant and unnecessary information.

Arguments can be represented as pyramids. The lower level holds non-fluent statements, and the level above holds statements about the initial situation. Higher levels hold statements derived from statements appearing on the lower levels. Beliefs in higher levels are intermediate results in the derivation of the overall claim. The derivations of these beliefs correspond to arguments, the collection of which form the main argument.

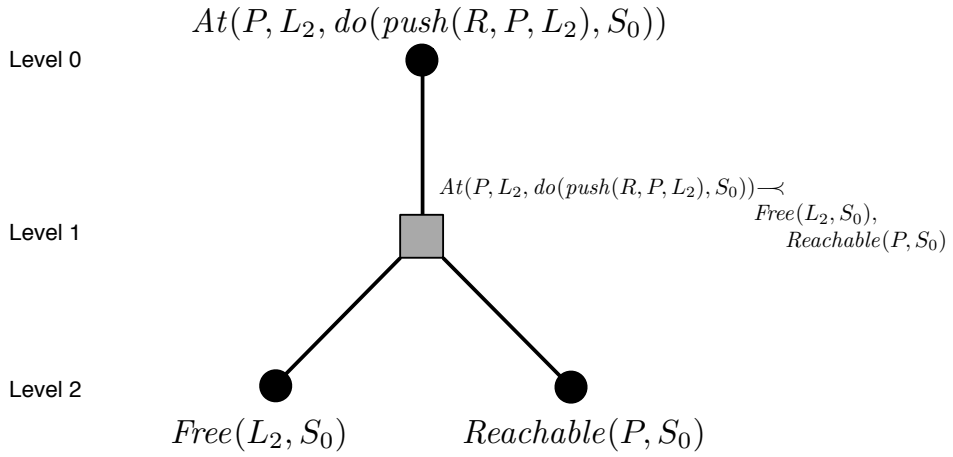
**Definition 19.** *An argument* $\langle \mathcal{B}', h' \rangle$ *is a* subargument *of* $\langle \mathcal{B}, h \rangle$ *if* $\mathcal{B}' \subseteq \mathcal{B}$.

**Example 1.** *The following argument claims that a parcel P is in a location* $L_2$ *after being pushed to this location, because the location is free and the parcel is reachable. The argument has been constructed using the axiom*

$$At(p,l,do(a,s)) \prec\!\!\!- a = push(p,l), Free(l,s), Reachable(p,s); At(p,l,s), a \neq push(p,l')$$

*and the ground initial situation presumption* $Free(L_2, S_0) \prec\!\!\!-$ *and* $Reachable(P, S_0) \prec\!\!\!-$.

$$\langle \{At(P,L_2,do(push(R,P,L_2),S_0)) \prec\!\!\!- Free(L_2,S_0), Reachable(P,S_0), \\ Free(L_2,S_0) \prec\!\!\!-,\ Reachable(P,S_0) \prec\!\!\!-\}, \\ At(P,L_2,do(push(R,P,L_2),S_0))\rangle$$




*This argument can be represented as the following tree:*
*The tree is organised in levels. The top level is level* 0*, which contains the argument's claim. Odd-numbered levels contain defeasible rules. Even-numbered levels represent the literals that are used to derive the argument's claim. The literals on the same level refer to the same situation. The leafs of the tree are beliefs derived using presumptions or default negated literals.*

#### 3.3.6.2 Attacks

Arguments generated from a contradictory theory may present contradicting claims. Such arguments are linked through the *attack relation*. The definition of the attack relation is based on the notion of a sub-argument.

**Definition 20.** *The argument* $\langle \mathcal{B}_1, h_1 \rangle$ attacks $\langle \mathcal{B}_2, h_2 \rangle$ *at literal* $h_3$ *iff there exists a subargument* $\langle \mathcal{B}_3, h_3 \rangle$ *of* $\langle \mathcal{B}_2, h_2 \rangle$ *such that* $h_1$ *is the negation of* $h_3$.

The attack relation between the attacker and the attacked sub-argument is symmetric. It provides no grounds for the resolution of ties between arguments attacking each other. Consider, for instance, two arguments claiming contradicting beliefs about the initial state. These arguments mutually attack each other. If there exist no other attackers to resolve this tie, the conflict cannot be resolved. As a result, depending on the employed argumentation semantics we need to either accept both or none of the claims. Therefore, the result is either having to deal with uncertainty in the planning domain, or synthesise weak plans whose success is suggested by contradictory evidence.

In order to overcome this issue, we inspect the internal structure of arguments, while trying to identify whether there are grounds for preferring one over the other.

Relying on a preference ordering over arguments is a common approach in the literature (Prakken and Sartor, 1997; Amgoud and Cayrol, 1998). Alternative approaches have also been proposed for aggregating information about preferences (Amgoud et al., 2000b; Brewka, 2001).

The *defeat* relation builds on the attack relation by taking the preference of the conflicting arguments into account. Informally, an argument defeats another argument if the first argument attacks the second argument, and has a higher preference than the sub-argument that is directly attacked.

**Definition 21.** *Argument b* defeats *argument $b'$ iff:*

- *b attacks $b'$ at its sub-argument $b''$, and*
- $pref(b) \geq pref(b'')$.

In order to take preference orderings into account, we compare the preference value of the attacker with the one of the sub-argument that is directly attacked. The reason for this is that the preference of the complete argument is determined by its other sub-arguments as well, which are irrelevant to the attack. Consider an argument that makes a claim about a literal after applying to actions in the initial situation. Also, assume that the claim is supported by an initial situation belief and that there is an attack exactly against this belief. It is obvious that in order to determine whether the attacker defeats the argument we only need to take information regarding the initial situation into account, since anything related to future situations should not bias the strength of the initial situation belief that is attacked.

The notion of argument preference is used in the literature as a tie-breaking mechanism, when there is no reason to prefer one argument over another. Multiple ways have been proposed to provide the means for measuring argument preference (e.g. *generalised specificity* (Simari and Loui, 1992), the *weakest link* (Pollock, 2001), and the *last link* (Prakken and Sartor, 1997) principles). These are usually based on structural characteristics of the arguments, as for example the number of derivation steps necessary to reach the argument's claim, or on preference orderings over the beliefs that support these arguments. The specification of the preference ordering mechanism is usually a domain specific way to fine-tune the system.

Our methods are generic with respect to a specification of the preference ordering among arguments. In the following analysis, we outline a simple definition of preference relation based on a preference ordering over beliefs and the weakest link

principle. If such information is not available, the defeat relation is equivalent to the attack relation.

**Definition 22.** *Let pref be a function which given a belief returns an integer, and assume that it is instantiated for every belief in the knowledge base. The preference value of an argument a is the lowest preference value of a belief in its support set,* $pref(a) = \min_{\phi \in Support(a)} pref(\phi)$.

More elaborate preference calculation mechanisms can be developed by focusing on domain specific characteristics of individual theories. For instance, if an agent is aware that the frame parts of the successor state axioms in the theory do not correctly reflect the actions responsible for changing the value of the literal, it is useful to fine-tune the defeat relations so that conclusions reached through the effect part of axioms are preferred to conclusions reached using the frame part of the axioms.

Usually, preference values are used to represent the credibility of the beliefs. Depending on the modelled domain, these values may denote (or aggregate) notions like:

- Authority: In situations in which different agents have different roles.
- Capabilities: For instance, the preference values of axioms regarding the effects of an action are higher for the agent executing this action.
- Number of conditions: More refined axioms are more credible. For example, the more conditions on a conditional effect, the more credible the axiom, under the rationale that agents without specialised knowledge may hold a generic specification of the action.
- Timestamps on beliefs coming from observations: The newer the better.
- Learning from experience: Past execution failures reduce credibility.
- Preference over types of axioms: For example, derivations made using the effect part are stronger than those made using the frame part.

We assume that the agents agree on the way preference orderings are calculated (e.g. weakest link or generalised specificity). However, we do not consider that belief preference values are shared. Nevertheless, agents must accept the preference values presented by their peers. This assumption follows from the cooperative nature of MPCP. For example, consider a scenario in which agent $i$ has reasons to beliefs that $Light(L_1, S_0)$ holds with a preference value of $x$, and agent $j$ has reasons to believe that

the same predicate holds, and has a preference value of $y$ for this belief. Assume that in this case preferences are based on the time an observation was made. Both agents accept the information of their peers, and form two arguments claiming $Light(L_1, S_0)$, with different preference values $x$ and $y$. Obviously, removing the argument with the lowest preference will not affect the warrant results that are reached from the theory. Therefore, maintaining both arguments is not necessary in practice, since agents can always only use the highest preference value that is available for a believe.

#### 3.3.6.3 Warrants

Argumentation theory provides theoretical tools for defining the notion of acceptability in terms of arguments. *Abstract argumentation* methods define concrete acceptability semantics by looking at arguments at an abstract level, disregarding their internal structure, and focusing exclusively on the defeat relations between them.

Different argumentation semantics (Baroni and Giacomin, 2009) can be used in conjunction with our framework to define the notion of acceptability. Here, for simplicity, we employ grounded (sceptical) acceptability semantics. Grounded semantics impose a strict notion of acceptability to statements and plans that are warranted from a domain theory, since it requires every acceptable argument to be defended by a set of arguments, which does not include itself.

With this, we define the notion of *warrant* for ground literals. The notion of warrant is twofold, it requires the existence of a defeasible derivation that forms an argument, and that this argument is defended against every potential defeat.

**Definition 23.** *Suppose a DBAT $\mathcal{D}$ and the corresponding argumentation framework $AF = \langle Args, Defs \rangle$, for every argument that can be constructed from ground$(\mathcal{D}, S)$, for any ground situation term $S$. Let $GE_{AF}$, the grounded extension of $AF$. Any ground literal $L(S)$ is* warranted *from ground$(\mathcal{D}, S)$, denoted ground$(\mathcal{D}, S) \mid\!\approx L(S)$, if and only if there exists an argument $A \in Args$, with $Claim(A) = L(S)$, such that $A \in GE_{AF}$.*

Note that in the special case that $L$ is a non-fluent symbol, the grounding is preformed only for situation $S_0$.

The number of ground situation terms is infinite as each situation represents a different history (i.e. sequence of actions). Therefore, even if there is a finite number of actions in the domain, grounding the theory for all situations will result in an infinite set. If we focus on a grounded situation term the ground theory remains finite if the axioms and the objects in the domain are finite.

Similar to defeasible derivations, the warrant results in DBATs are not affected by grounding the theory with respect to successor situation terms. This proposition asserts that warrant results for a plan $\pi$ are also relevant to every plan extending $\pi$.

**Proposition 5.** *Assume a defeasible basic action theory $\mathcal{D}$, and two ground situation terms $S_k$ and $S_l$ such that $S_k$ is predecessor of $S_l$. For every ground situation term $S_i$ and every ground fluent literal $L$, if $ground(\mathcal{D}, S_k) \mid\!\approx L(S_i)$ then $ground(\mathcal{D}, S_l) \mid\!\approx L(S_i)$.*

*Proof.* Definition 16 states that all ground defeasible basic action theories include all axioms grounded for all combinations of ground terms that respect the expressed equalities and inequalities. The ground theories $ground(\mathcal{D}, S_l)$ and $ground(\mathcal{D}, S_k)$ have been grounded with exactly the same constants, apart from the additional situation terms that are successor to $S_k$ and predecessor or equal to $S_l$. Therefore, $ground(\mathcal{D}, S_k) \subset ground(\mathcal{D}, S_l)$. $ground(\mathcal{D}, S_k) \mid\!\approx L(S_i)$ entails that there exists an argument $\alpha$ claiming $L(S_i)$ and that this argument can be defended against every defeater. Every argument that can be constructed from $ground(\mathcal{D}, S_k)$ can also be constructed from $ground(\mathcal{D}, S_l)$. Also, every argument that can be constructed from $ground(\mathcal{D}, S_l)$, but not from $ground(\mathcal{D}, S_k)$, does not defeat $\alpha$, since its conclusion refers to a situation that is successor to $S_k$. As as result, if $ground(\mathcal{D}, S_k) \mid\!\approx L(S_i)$ then $ground(\mathcal{D}, S_l) \mid\!\approx L(S_i)$. □

For readability purposes, we write $\mathcal{D} \mid\!\approx L(S)$ to represent $ground(\mathcal{D}, S) \mid\!\approx L(S)$. Overloading of the warrant does not lead to inconsistencies, since every other potential grounding extending $ground(\mathcal{D}, S)$ leads to the same results.

**Proposition 6.** *Assume a defeasible basic action theory $\mathcal{D}$, and a ground situation term $S$. If $ground(\mathcal{D}, S) \mid\!\approx L(S)$, there does not exist $S'$ such that $ground(\mathcal{D}, S) \cup ground(\mathcal{D}, S') \mid\!\not\approx L(S)$.*

*Proof.* Proof by contradiction. Assume that there exists $S'$ such that $ground(\mathcal{D}, S) \cup ground(\mathcal{D}, S') \mid\!\not\approx L(S)$. Let the set $Args_{L(S)}$ be the subset of the arguments that can be constructed from $ground(\mathcal{D}, S)$, such that it contains every argument claiming $L(S)$, their defeaters, the defeaters of their defeaters, etc. There exists no argument that can be constructed from the beliefs in $ground(\mathcal{D}, S')$, which does not already exist in $Args_{L(S)}$, and defeats an argument in $Args_{L(S)}$. This is the case, since every belief from $ground(\mathcal{D}, S')$ that does not exist in $ground(\mathcal{D}, S)$ refers to a situation term that does not appear in the arguments in $Args_{L(S)}$. As a result, $ground(\mathcal{D}, S')$ is irrelevant to the warrant status of $L(S)$. □

Grounded semantics assert that the warrant relation is contradiction-free.

**Proposition 7.** *Suppose a defeasible action theory $\mathcal{D}$ and a ground predicate $L(S)$. If $\mathcal{D}|\!\approx L(S)$ then $\mathcal{D}|\!\not\approx \overline{L}(S)$.*

*Proof.* If $\mathcal{D}|\!\approx L(S)$ and $\mathcal{D}|\!\approx \overline{L}(S)$, there exist two arguments $a_{L(S)}$ and $a_{\overline{L}(S)}$ with $Claim(a_{L(S)}) = L(S)$ and $Claim(a_{\overline{L}}) = \overline{L}(S)$, which are both part of the grounded extension of the argumentation framework containing all arguments that can be constructed from $ground(\mathcal{D}, S)$. This is impossible, since the grounded extension of an argumentation framework is conflict-free (Dung, 1995). □

The notion of warrant is extended to conjunctive statements. For any ground literal predicates $L_1, L_2, \ldots, L_n$ we write:

$$\mathcal{D}|\!\approx L_1, L_2, \ldots, L_n \text{ if and only if } \mathcal{D}|\!\approx L_1, \mathcal{D}|\!\approx L_2, \ldots, \text{ and } \mathcal{D}|\!\approx L_n\,.$$

This is essential for formalising the notion of plan acceptability, which entails that every action of the plan is executable and that the goal is achieved. An alternative solution without extending the warrant relation would require the addition of rules specifying the requirements for the acceptability of a plan. Due to the limitations of the expressive power of the formalism, this would need to be done for every situation and goal.

#### 3.3.6.4 Warranted Plans

A warranted plan is a sequence of actions if the beliefs that every action can be executed in sequence and that the goal is achieved in the resulting situation are warranted.

**Definition 24.** *Suppose a defeasible action theory $\mathcal{D}$, and let $AF = \langle Args, Defs\rangle$ be the argumentation framework for all arguments that can be constructed from $\mathcal{D}$, and $GE_{AF}$ its grounded extension. Let the expression $G_1, \ldots, G_m$ be the shared goal of the agents. The sequence of actions $\pi = A_1, A_2, \ldots, A_n$ is a* warranted plan *if and only if:*

$$\mathcal{D}|\!\approx Poss(A_1, S_0), Poss(A_2, S_1), \ldots, Poss(A_n, S_{n-1}), G_1(S_n), \ldots, G_m(S_n)\,,$$

*where $S_i = do(A_1, S_{i-1})$ denotes the situation resulting from the application of action $A_i$ to the predecessor situation $S_{i-1}$.*

We use terms warranted and acceptable plan interchangeably.

## 3.4 Axiomatising Planning Domains

This section discusses important issues related to the process of encoding planning domains in the form of DBATs. We focus on the internal structure of defeasible successor state axioms that is required to represent planning operators, and explain how MPCP problems can be expressed as DBATs.

### 3.4.1 Encoding Defeasible Successor State Axioms

Basic action theories restrict the form of axioms, especially with respect to the situation terms that appear within the axioms. However, they do not impose a specific internal structure on the bodies of the rules. In order to axiomatise planning domains as DBATs, we need to impose additional structure. This structure must account for both the effect, and the frame information that describes under which conditions literals remain unaffected by the application of actions. Successor state axioms are compound rules that incorporate defeasible *effect* and *frame* axioms.

#### 3.4.1.1 Defeasible Effect Axioms

We describe how effect axioms are encoded using the following example. Consider the operator $switch\_on(l)$, with the conditional effect $\langle\{Plugged(l)\}, Lit(l)\rangle$. This effect states that we have reasons to believe that a lamp is lit, after performing the *switch_on* operator, if we have reasons to believe that it is plugged to a power source. This effect is encoded by the following defeasible rule:

$$Lit(l, do(a,s)) \multimap a = switch\_on, Plugged(l,s).$$

Multiple conditional effects from different specifications that produce the same effect literal are written within the same axiom using disjunction. The following axiom adds an effect from a different specification stating that we have reason to believe that a lamp is lit if, after performing the *switch_on* operator, if we have reasons to believe that it is not broken:

$$Lit(l, do(a,s)) \multimap a = switch\_on, Plugged(l,s); a = switch\_on, \sim Broken(l,b,s).$$

Obviously, without additional domain-specific knowledge, the agents cannot reach the conclusion that the specification of both effects is incomplete, and a complete specification can be achieved by combining the two.

Every conditional effect for an operator $A$ of the form $\langle\{C_1, C_2, \ldots, C_n\}, L\rangle$ in an agent's specification corresponds to a defeasible effect axiom of the form:

$$L(do(a,s)) \prec\!\!- a = A, C_1(s), C_2(s), \ldots, C_n(s)\,.$$

Defeasible effect axioms that produce the same literal can be combined in a single axiom: $L(do(a,s)) \prec\!\!- \gamma_L(s)$, where $\gamma_L(s)$ abbreviates the disjunction of the bodies of all relevant effect axioms. For the previous example, we write $Lit(l, do(a,s)) \prec\!\!- \gamma_{Lit}(l,s)$, where $\gamma_{Lit}(l,s)$ abbreviates the expression $a = switch\_on, Plugged(l,s); a = switch\_on, \sim Broken(l,b,s)$.

#### 3.4.1.2 Defeasible Frame Axioms

Constructing a defeasible frame axiom is a more complicated process. The frame axiom encodes that we have reasons to believe that a literal holds after the application of an action $a$ in a situation $s$, if it holds in the previous situation, and every conditional effect producing its complements is inapplicable. Consider for instance the following frame axiom produced for the first specification of the previous example:

$$\sim Lit(l, do(a,s)) \prec\!\!- \sim Lit(l,s), not(a = switch\_on, Plugged(l,s)).$$

This axiom states that we have reasons to believe that the lamp is not lit after the application of $a$ in $s$, if we have indications that it is not lit in $s$ and it is not the case that $a$ is the action of switching on the lamp or $l$ is not plugged in the power source. Accordingly, the frame axiom for the second operator specification is the following:

$$\sim Lit(l, do(a,s)) \prec\!\!- \sim Lit(l,s), not(a = switch\_on, \sim Broken(l,b,s)).$$

We encode one frame axiom for each operator specification, since every specification implicitly states not only what changes due to the application of the action, but also what remains the same. All effect and frame axioms referring to the same fluent literal are combined within one successor state axiom. This axiom needs to be transformed to disjunctive normal form following the specification of defeasible rules.

The frame axiom for a literal $L$ has the following form:

$$L(do(a,s)) \prec\!\!- L(s), not(\gamma_{\overline{L}}(s))\,,$$

where $\gamma_{\overline{L}}(s)$ abbreviates the body of the compound effect axiom that is encoded from the agent's theory for the complement of literal $L$.

#### 3.4.1.3 Well Formed Defeasible Frame Axioms

The structure of defeasible axioms only allows default negation immediately preceding fluent literals. This is not the case in the aforementioned form as $\gamma_L(s)$ may consist of multiple effects and conditions. To produce a well-formed axiom we transform the axioms in a fashion similar to the Lloyd-Topor rules (Reiter, 2001). We present the axiom $L \prec body$, in the form $L \prec \psi, \phi, \psi'$, where any one of $\psi$ and $\psi'$ may be missing.

1. Replace $L \prec \psi, not(\phi_2; \phi_3), \psi'$ by
$$L \prec \psi, not\, \phi_2, not\, \phi_3, \psi'.$$
2. Replace $L \prec \psi, not(\phi_2, \phi_3), \psi'$ by
$$L \prec \psi, (not\, \phi_2; not\, \phi_3), \psi'.$$
3. Replace $L \prec \psi, not(a = A), \psi'$ by
$$L \prec \psi, a \neq A, \psi'$$
4. Replace $L \prec \psi, (\phi_2; \phi_3), \psi'$ by
$$L \prec \psi, \phi_2, \psi' \text{ and } L \prec \psi, \phi_2, \psi'.$$
5. Replace every rule of the form $L \prec \phi_i$, where $\phi_i$ is a conjunction of (possibly default negated) literals, equalities and inequalities with the rule $L \prec \bigwedge_{\forall i} \phi_i$.

The first two transformations assert that the default negation symbol appears only before fluent literals. The third rule simplifies the resulting axiom from disjunctions. Its application generates a set of axioms for each head $L$, whose body consists of a conjunction of literals. The final rule integrates these into a single defeasible axiom.

The axiom resulting after the application of these rules to an axiom of the form $L(do(a,s)) \prec L(s), not(\gamma_L(s))$ is a well formed defeasible axiom. This is the case, since default negation appears in the resulting axiom only before fluent literals, and the body of the rules is either a conjunction of literals, equalities and inequalities, or a disjunction of multiple conjunctions, equalities and inequalities.

For effects with at most one condition, the application of the transformation is straightforward. Consider the frame axiom from the previous example:

$$\sim Lit(l, do(a,s)) \prec \sim Lit(l,s), not((a = switch\_on, Plugged(l,s)); \\ (a = switch\_on, \sim Broken(l,b,s)).$$

After the application of the first transformation rule we have:

$$\sim Lit(l, do(a,s)) \multimap \sim Lit(l,s), (not(a = switch\_on, Plugged(l,s)), \\ not(a = switch\_on, \sim Broken(l,b,s)).$$

We apply the second transformation and rewrite the expression as follows:

$$\sim Lit(l, do(a,s)) \multimap \sim Lit(l,s), (not(a = switch\_on); not\, Plugged(l,s)), \\ (not(a = switch\_on); not \sim Broken(l,b,s)).$$

The third rule transforms default negated equalities to inequalities:

$$\sim Lit(l, do(a,s)) \multimap \sim Lit(l,s), (a \neq switch\_on; not\, Plugged(l,s)), \\ (a \neq switch\_on, not \sim Broken(l,b,s)).$$

The application of the fourth rule simplifies the axiom:

$$\sim Lit(l, do(a,s)) \multimap \sim Lit(l,s), a \neq switch\_on, a \neq switch\_on$$
$$\sim Lit(l, do(a,s)) \multimap \sim Lit(l,s), a \neq switch\_on, not \sim Broken(l,b,s)$$
$$\sim Lit(l, do(a,s)) \multimap \sim Lit(l,s), not\, Plugged(l,s), a \neq switch\_on$$
$$\sim Lit(l, do(a,s)) \multimap \sim Lit(l,s), not\, Plugged(l,s), not \sim Broken(l,b,s).$$

Finally, the fifth rule combines the above into one frame axiom:

$$\sim Lit(l, do(a,s)) \multimap \sim Lit(l,s), a \neq switch\_on, a \neq switch\_on; \\ \sim Lit(l,s), a \neq switch\_on, not \sim Broken(l,b,s); \\ \sim Lit(l,s), not\, Plugged(l,s), a \neq switch\_on; \\ \sim Lit(l,s), not\, Plugged(l,s), not \sim Broken(l,b,s).$$

The resulting frame axiom for $\sim Lit$ requires that either none of the actions producing *Lit* applied, or that if one is applied, at least one of its conditions are not warranted.

**Observation 1.** *Let an action A that produces L under conditions $C$. Either $a \neq A$ or $\sim C$, with $C \in \mathcal{C}$, appears in every disjunct in the body of the frame axiom for $\overline{L}$.*

#### 3.4.1.4 Default Negation in Defeasible Frame Axioms

The frame axiom specification is based on default negation preceding the conditions of the contradicting effects. If we use normal negation instead, we need to be able to

derive all conditions in the body of the rule in order to make a derivation using this frame axiom. For instance, consider the first frame axiom from the previous example, grounded for lamp $l = L_1$ and situation $S_1 = do(switch\_on, S_0)$:

$$\sim Lit(L_1, S_1) \prec\!\!- \sim Lit(L_1, S_0), \sim Plugged(L_1, S_0).$$

If there is no defeasible derivation for $\sim Plugged(L_1, S_0)$, then we cannot construct a defeasible derivation for $\sim Lit(L_1, S_1)$ using this frame axiom. This situation causes problems when there is *uncertainty* about the domain. If for example $\mathcal{D}_{S_0} = \{\sim Lit(L_1, S_0) \prec\!\!-\}$, and $\mathcal{D}_{ss}$ contains the first specification from the above example, then $\mathcal{D} \not|\sim \sim Lit(L_1, S_1)$ and $\mathcal{D} \not|\sim Lit(L_1, S_1)$, since both rules are inapplicable as $\{Plugged(L_1, S_0) \prec\!\!-, \sim Plugged(L_1, S_0) \prec\!\!-\} \cap S_0 = \emptyset$.

On the contrary, consider the same frame axiom with default negation preceding contradictory effects, grounded for lamp $l = L_1$ and situation $S_1 = do(switch\_on, S_0)$:

$$\sim Lit(L_1, S_1) \prec\!\!- \sim Lit(L_1, S_0), not\, Plugged(L_1, S_0).$$

If $\mathcal{D} \not|\sim \sim Lit(L_1, S_0)$ then $\mathcal{D} \not|\sim \sim Lit(L_1, S_1)$, since following the semantics of default negation in DeLP, default negated literals are treated as assumptions. The burden of evaluating these assumptions is shifted to the argumentation phase. Essentially, if there exists an undefeated argument claiming $Plugged(L_1, S_0)$ the assumption is refuted. Otherwise, and if $\mathcal{D} |\!\approx \sim Lit(L_1, S_0)$, then $\mathcal{D} |\!\approx \sim Lit(L_1, S_1)$.

#### 3.4.1.5 Defeasible Successor State Axioms

For every operator specification and fluent literal, we construct one defeasible successor state axiom, by combining the effect and frame axioms described above. For literal $L$ this takes the following form:

$$L(do(a, s)) \prec\!\!- \gamma_L(s); \phi_{\overline{L}}(s)\,,$$

where $\phi_{\overline{L}}(s)$ denotes the body of the well formed defeasible frame axiom after the application of the necessary transformations.

### 3.4.2 Encoding MPCP Problems as DBATs

In this section, we consider how derivations made from MPCP problems relate to inferences from DBATs. In the general case, the two formalisms do not necessarily lead

to equivalent conclusions. The major difference between these reasoning mechanisms is outlined by a special case illustrated in the following example.

Consider a MPCP problem with two operators *plug* and *switch_on*, such that the first has the effect $\langle\{\neg Cut\}, Plugged\rangle$ and the second has the effect $\langle\{Plugged\}, Lit\rangle$. Let the initial state $I = \{Cut, \neg Cut, \neg Lit\}$. Projecting the plan $\langle plug, switch\_on\rangle$ leads to calculating the following states:

$$\sigma_1 = \gamma(plug, I) = \{Cut, \neg Cut, \neg Lit, Plugged\}, \text{ and}$$

$$\sigma_2 = \gamma(switch\_on, \sigma_1) = \{Cut, \neg Cut, Plugged, Lit\}.$$

The uncertainty regarding the predicate *Plugged* in the initial state leads to having reasons to believe that *Plugged* holds in $\sigma_1$. However, since $\neg Plugged \notin I$, we do not have reasons to believe that $\neg Plugged$ holds in $\sigma_1$, even though we have reasons to believe that the effect of the operator *plug* may not be applicable. As a result $\neg Lit \notin \sigma_2$.

The specification of successor state axioms described in the previous section treats uncertainty differently. Consider the same domain written as the DBAT $\mathcal{D}$. The set of initial situation axioms contains the rules: $Cut{-\!\prec}$, $\sim Cut{-\!\prec}$, and $\sim Lit{-\!\prec}$. The following successor state axioms are encoded:

$$Plugged(do(a,s)){-\!\prec} a = plug, \sim Cut(s),$$

$$Lit(do(a,s)){-\!\prec} a = switch\_on, Plugged(s), \text{ and}$$

$$\sim Lit(do(a,s)){-\!\prec} \sim Lit, not(a = switch\_on, Plugged(s)).$$

For the derivation of $\sim Lit(S_2)$, where $S_2 = do(switch\_on, S_1)$ and $S_1 = do(plug, S_1)$ we construct the following arguments:

$$\alpha = \langle\{\sim Lit(S_0){-\!\prec}, \sim Lit(S_1){-\!\prec} \sim Lit(S_0), \sim Lit(S_2){-\!\prec} \sim Lit(S_1),\ not\, Plugged(S_1)\}, \sim Lit(S_2)\rangle,$$

$$\beta = \langle\{\sim Cut(S_0){-\!\prec}, Plugged(S_1){-\!\prec} \sim Cut(S_0)\}, Plugged(S_1)\rangle, \text{ and}$$

$$\alpha' = \langle\{Cut(S_0){-\!\prec}\}, Cut(S_0)\rangle.$$

The argument $\beta$ attacks the assumption $not\, Plugged(S_1)$ of argument $\alpha$, and argument $\alpha'$ attacks the premises $\sim Cut(S_0)$ of argument $\beta$. If $\alpha'$ has higher preference than $\beta$, then $\alpha$ is acceptable and $\mathcal{D} |\!\approx \sim Lit(S_2)$.

The above example shows a case in which the defeasible mechanism leads to inferences that cannot be made by the set-theoretic notation. This situation arises due

to the initial uncertainty with respect to the predicate *Plugged*. The defeasible reasoning mechanism using the default negation makes the assumption $not\,Plugged(S_1)$, although $\mathcal{D} \mid\sim Plugged(S_1)$ and $\mathcal{D} \mid\not\sim \sim Plugged(S_1)$. Then, since the argument attacking the assumption is itself defeated, the assumption is defended and argument $\alpha$ is acceptable.

#### 3.4.2.1 MPCP with Incomplete Initial States

The problematic situation illustrated by the previous example is caused by the interaction of incompleteness and ambiguity. In order to be able to overcome this issue we transform incompleteness to ambiguity. This can be accomplished in two ways: changing the planning theory, or altering the state transition function.

The MPCP problem with incomplete initial states can be transformed to problems with complete initial states by simply adding to the initial state $I$ both literals $L$ and $\overline{L}$ for every ground literal $L$, if neither $L$ or $\overline{L}$ is part of $I$. If preference-based measures are used, we assign a minimum preference value to the introduced literals.

Alternatively, we can utilise an alternative state transition function specification which transforms incompleteness to ambiguity. The problematic situations arise when we calculate the successor state, for literals which we have no information about. For example, consider the effect $\langle C, L\rangle$ of an action $A$. If both $\overline{L}$ and $L$ are not part of state $\sigma$, then, if there exists at least one uncertain condition in $C \in \overline{C}$ such that both $C$ and $\overline{C}$ is part of $\sigma$, we must introduce both $L$ and $\overline{L}$ to $\gamma(A,\sigma)$. $L$ is introduced due to the conditional effect, whereas $\overline{L}$ is introduced because the applicability of the effect is ambiguous and there is uncertainty regarding $L$ and $\overline{L}$ in $\sigma$.

Altering the state transition function leads to a specification that is significantly different from the "classical" definition. As a result, and in order to be able to utilise already existing systems with minor modifications, we focus on translating incompleteness in the initial states of the planning domain theory to ambiguity.

#### 3.4.2.2 Initial State Completeness Assumption

We introduce the assumption that there is no incompleteness in the initial state of the planning domain. This does not imply that the initial state is unambiguous.

**Assumption 1.** *Let a MPCP problem $P = \langle N, F, I, O, G\rangle$. The initial state $I$ is* complete *if for every ground literal $L$ in $F_c$, it holds that $\{L, \overline{L}\} \cap I \neq \emptyset$.*

If there is no incompleteness in the initial state of the planning problem, then there is no incompleteness in any state of domain.

**Proposition 8.** *Consider a MPCP problem $P = \langle N, F, I, O, G \rangle$, and let there be no incompleteness in the initial state $I$. It is always the case that $L \in \sigma_k$ or $\overline{L} \in \sigma_k$, where $\sigma_k$ is the state reached by a plan $\langle a_1, \ldots, a_k \rangle$.*

*Proof.* Proof by induction on the length of the plan $k$.
(Base case) $k = 0$. Holds from proposition hypothesis.
(Induction step) We assume that it holds for $k = n$ and we need to show that it holds for $k = n + 1$.
Proof by contradiction. We assume that for some literal $L$:

$L \notin \sigma_{n+1}$ (A) and $L \notin \sigma_{n+1}$ (B).

From (A) $\Rightarrow L \notin \sigma_n$ (A1) and there is no applicable effect producing $\overline{L}$ (A2) or there is no applicable effect producing $L$ (A3).

From (B) $\Rightarrow \overline{L} \notin \sigma_n$ (B1) and there is no applicable effect producing $L$ (B2) or there is no applicable effect producing $\overline{L}$ (B3).

So (A) and (B) $\Rightarrow$ ((A1) and (A2) or A3) and ((B1) and (B2) or B3) $\Rightarrow$

(A1 and A2 and B1 and B2) (i) or (A3 and B1 and B2) (ii) or (A1 and A2 and B3) (iii) or (A3 and B2) (iv).

If (i) then A1 $\Rightarrow L \notin \sigma_n$ and B2 $\Rightarrow \overline{L} \notin \sigma_n$, which refutes the derivation step.

(ii) is impossible since A3 and B2 contradict each other.

(iii) is impossible since A2 and B3 contradict each other.

If (iv) then from the induction step's assumption $L \in \sigma_n$ or $\overline{L} \in \sigma_n$. On the one hand, if $L \in \sigma_n$, from B3 we have that $L \in \sigma_{n+1}$. On the other, if $\overline{L} \in \sigma_n$, from A3 we have that $\overline{L} \in \sigma_{n+1}$. In any case, $L \in \sigma_{n+1}$ or $\overline{L} \in \sigma_{n+1}$, contradicting the assumption $L \notin \sigma_{n+1}$ (A) and $L \notin \sigma_{n+1}$.

□

We can exploit the initial state completeness assumption to simplify the definition of the individual agent's state transition function $\gamma_i$. The second condition of $\gamma_i$ details that a fluent literal is unaffected by the application of an action, if for every conditional

effect producing the complement of this literal, there exists a condition $C$ of this effect that is either not part of the state, or that its complement $\overline{C}$ is part of the state.

Due to the completeness of the initial state, every successor state is also complete. As a result, for every condition $C$, if $C$ is not part of the state $\sigma$, we can infer that $\overline{C} \in \sigma$. Accordingly, we simplify the individual state transition function specification when used in conjunction with domains without initial state incompleteness as follows:

**Definition 25.** *Consider agent i and the individual planning problem $P_i = \langle F, I_i, O_i, G \rangle$. Let a ground action $\langle pre_i(a), a, eff_i(a) \rangle \in ground(O_i)$ for which there is evidence suggesting that it is applicable in state $\sigma$. The state transition function $\gamma$ must assert that $e \in \gamma(a, \sigma)$ if and only if:*

1. *there exists an effect $\langle C, e \rangle \in eff_i(a)$ and $C \subseteq \sigma$, or*
2. *$e \in \sigma$ and for every conditional effect $\langle C, \overline{e} \rangle \in eff_i(a)$, there exists $c \in C$ such that $\overline{c} \in \sigma$.*

#### 3.4.2.3 Translation Mechanism

This section describes a translation mechanism for encoding multi-perspective cooperative planning problems in the form of defeasible basic action theories. Consider a MPCP problem $P = \langle N, F, I, O, G \rangle$.

The objects and the variables of the planning domain correspond to the ground and unground terms of sort object of the defeasible basic action theory.

$$\mathcal{D}_c = L_c$$

Planning operator names correspond to action functions in basic action theories.

$$\mathcal{D}_{una} = \{o = o \,|\, \langle pre, o, eff \rangle \in O\} \cup \{o_1 \neq o_2 \,|\, \langle pre, o_1, eff \rangle, \langle pre, o_2, eff \rangle \in O \text{ and } o_1 \text{ has a different name or arity from } o_2\}$$

Planning domain predicates correspond to fluents in the basic action with an additional final argument of sort situation. For instance, the planning domain predicate $Light(loc)$ corresponds to the fluent predicate $Light(loc, s)$. For simplicity we do not differentiate between fluent and non-fluent predicates, since this distinction is not explicit in MPCP problems. If the complexity of the domain demands this distinction, we can identify non-fluent predicates by inspecting the operators and searching for predicates that do not appear in the effects of any action.

The initial state axioms correspond to the initial state of the planning domain. Every fluent that holds in the initial state corresponds to a predicate that holds in the initial situation.

$$\mathcal{D}_{S_0} = \{L[S_0] \prec \mid L \in I\}$$

We use notation $\phi[s]$ to denote the formula obtained from substituting all fluent predicates $L(\vec{x})$ appearing in $\phi$ with predicate $L(\vec{x},s)$, which has $s$ as an additional final argument. Also, in order to account for the different notation, the negation symbol $\neg$ is replaced by the symbol $\sim$.

Defeasible action precondition axioms are constructed from the preconditions defined in the action specifications. For every operator $o$ we create one action precondition axiom. The left-hand side of the axiom is the predicate $Poss(o,s)$. The right hand side of the axiom comprises of a sequence of disjuncts, with each disjunct being the conjunction of the preconditions that are specified in a single agent's specification of the operator:

$$Poss(o,s) \prec \bigvee_{\forall i \in N} \bigwedge_{\forall C \in pre_i(o)} C[s] \,.$$

For example, given two specifications of the action $switch\_on(l)$ for agents 1 and 2 such that $pre_1(switch\_on(l)) = \{reachable(l), \neg broken(l)\}$ and $pre_2(switch\_on(l)) = \{powered(l)\}$, we construct the following defeasible action precondition axiom:

$$Poss(switch\_on(l),s) \prec reachable(l,s), {\sim}broken(l); powered(l) \,.$$

The set of defeasible action precondition axioms is generated as follows:

$$\mathcal{D}_{ap} = \{Poss(o,s) \prec \bigvee_{\forall i \in N} \bigwedge_{\forall C \in pre_i(o)} C[s] \;\mid\; \text{for every operator } o\} \,.$$

Defeasible successor state axioms have the structure discussed in Section 3.4.1:

$$L(do(a,s)) \prec \gamma_L(s); \phi_{\overline{L}}(s) \,,$$

Contrary to planning operator schemata, DBATs do not maintain a link between defeasible action precondition axioms and defeasible successor state axioms. As a result, if we have different specifications for the same action and one of them leads to the conclusion that the action is applicable, we infer that the resulting situation is executable. In order to be able to preserve the link between preconditions and effects in an operator specification we must be able to utilise only the successor state axioms that were created from the specifications which suggest that the action is applicable. To this end,

we incorporate the preconditions defined in every specification into the corresponding successor state axiom in the form of additional conditions in the operators' conditional effects. As a result, the effects are applicable, only in situations in which the action preconditions hold.

Let a MPCP problem $P$. The DBAT constructed using the above mechanism is called the *corresponding* DBAT for $P$. Next, we specify an important assumption that is necessary to illustrate the correctness of the translation mechanism.

#### 3.4.2.4 DBATs with Complete Initial States

The initial state completeness assumption carries over defeasible basic action theories. Consider a DBAT $\mathcal{D}$. $\mathcal{D}$ has a consistent initial state if for every ground fluent literal $L$, it holds that either $\mathcal{D}|\sim L(S_0)$ or $\mathcal{D}|\sim\overline{L}(S_0)$. The logical disjunction in the previous statement is not exclusive. It may be the case that both $\mathcal{D}|\sim L(S_0)$ and $\mathcal{D}|\sim\overline{L}(S_0)$ hold.

The following lemma outlines the effect of the initial state completeness assumption on defeasible derivations and warrants made with a DBAT whose initial situation rules contain only presumptions (i.e. rules of the form: $L(S_0)\multimap\prec$).

**Proposition 9.** *Consider a DBAT $\mathcal{D}$ with complete initial state whose initial situation rules contain only presumptions. Let $S_k$ a ground situation term, and L a fluent literal. If $\mathcal{D}|\not\approx L(S_k)$ then $\mathcal{D}|\sim\overline{L}(S_k)$.*

*Proof.* Proof by induction on situation term $S_k$.
(Base case) $k = 0$. From the hypothesis that the initial state is complete it holds that $\mathcal{D}|\sim L(S_0)$ or $\mathcal{D}|\sim\overline{L}(S_0)$. Since, $\mathcal{D}|\not\approx L(S_0)$, it is either the case that $\mathcal{D}|\not\sim L(S_0)$ or $\mathcal{D}|\sim L(S_0)$ and there is an argument defeating the argument corresponding to the derivation of $L(S_0)$. This is only possible if $\mathcal{D}|\sim\overline{L}(S_0)$. In any case, $\mathcal{D}|\sim\overline{L}(S_0)$.

(Induction step) We assume that it holds for $k = n$ and we need to show that it holds for $k = n+1$. Proof by contradiction.

> Assume that for every literal $L$ it holds that $\mathcal{D}|\not\approx L(S_{n+1})$ and $\mathcal{D}|\not\sim\overline{L}(S_{n+1})$ (1).
>
> From (1), it holds that $\mathcal{D}|\not\sim\overline{L}(S_n)$ (2) and for every effect rule $\overline{L}(S_{n+1})\multimap\prec\overline{C}_1(S_n),\ldots,\overline{C}_m(S_n)$ there exists $l\in\{1,\ldots,m\}$ such that $\mathcal{D}|\not\sim\overline{C}(S_n)$ (3).
>
> From (2) and the induction step, it holds that $\mathcal{D}|\approx L(S_n)$ (4).

From (4) and because warrant requires the existence of a corresponding derivation, it holds that $\mathcal{D}|\sim L(S_n)$. Also, the DBAT contains a successor state axiom for every literal, and the corresponding ground frame rule for literal $L$ has the form $L(S_{n+1}) \prec L(S_n), not\, C'_1(S_n), \ldots, not\, C'_1(S_{m'})$. Since a defeasible derivation does not require the derivation of literals preceded by default negation, from $\mathcal{D}|\sim L(S_n)$ we derive that $\mathcal{D}|\sim L(S_{n+1})$ (5).

From (3) we derive that there exists a ground frame axiom with head $L_{S_{n+1}}$ for which there is no attack on every condition in its body preceded by default negation. This is the case since every such axiom describes in its body a sequence of conditions, whose absence causes every effect producing the complement of the literal in the body of the frame axiom not to be applicable (6).

As a result, there exists an argument claiming $L(S_{n+1})$ (from (5)), whose premises cannot be defeated (from (4) and (6)) and whose conclusion cannot be defeated (1).

Therefore, this contradicts the assumption that $\mathcal{D}|\not\sim L(S_{n+1})$, proving the lemma. □

A corollary of the previous proposition is that for every such DBAT $\mathcal{D}$, literal $L$ and ground situation term $S$, it is either the case that $\mathcal{D}|\sim L(S)$ or $\mathcal{D}|\sim \overline{L}(S)$.

#### 3.4.2.5 Relating Candidate and Warranted Plans

This section connects the conclusions made using the defeasible and the set-theoretic planning formalisms. We show that every warranted plan is a candidate plan. This observation is useful in restricting the application of the complex argumentation process to candidate plans.

First of all, we show that defeasible derivations in defeasible basic action theories are strictly more relaxed than the corresponding derivations made using the state transition function. The reason behind this is the use of default negation in successor state axioms. Default negation is interpreted as an assumption, and shifts the burden of evaluating these assumptions to the argumentation mechanism. As a result, derivations with arbitrary assumptions, which are later disqualified in the argumentation phase, are made. Consider the following example.

**Example 2.** *The successor state axiom* $Lit(do(a,s)) \prec Lit(s), (a \neq switch\_off; a = switch\_off, not\, Working\_Switch(s))$ *describes an operator specification in which the*

*predicate Lit is negatively affected by flipping the light's switch, if the switch is functioning correctly. Assume that, for DBAT $\mathcal{D}$ and a ground situation S, we derive that $\mathcal{D}|\sim Working\_Switch(S)$ and $\mathcal{D}|\not\sim Working\_Switch(S)$. $\mathcal{D}|\sim Lit(do(switch\_off, S))$ is a valid derivation if $\mathcal{D}|\sim Lit(S)$. Obviously, the argument corresponding to the later derivation is defeated by a counterargument based on the derivation of $Working\_Switch(S)$.*

The state transition function $\gamma$ has an ambiguity propagating nature both with respect to inertia and effect conditions. Consider the previous example, and the case in which there is ambiguity regarding the predicate *Working_Switch* in a state $\sigma$ (meaning that $\{Working\_Switch, \neg Working\_Switch\} \subseteq \sigma$). There is also ambiguity in the successor state $\sigma'$ regarding the predicate *Lit* (meaning that $\{Lit, \neg Lit\} \subseteq \sigma'$). As a result, assumptions that will be defended in the argumentation phase with counterarguments attacking the defeating arguments' support, are also derived using the state transition function, exactly due to ambiguity in the support.

**Proposition 10.** *Consider a MPCP problem $P = \langle N, F, I, O, G\rangle$, and the corresponding DBAT $\mathcal{D} = \langle \mathcal{D}_{ss}, \mathcal{D}_{ap}, \mathcal{D}_{S_0}, \mathcal{D}_{una}, \mathcal{D}_c\rangle$. Also, let a possibly empty sequence of actions $\langle A_1, \ldots, A_k\rangle$. $\sigma_k = \gamma(\langle A_1, \ldots, A_k\rangle, I)$ is the state reached after performing these actions, and $S_k = do([A_1, \ldots, A_k], S_0)$ is the corresponding situation for the application of this action sequence. For every ground predicate literal L, if $L \in \sigma_k$, then it holds that $\mathcal{D}|\sim L(S_k)$.*

*Proof.* The proof is based on induction on the length of the plan $k$.

(Base case) $k = 0$. From the proposition hypothesis we have that $L(\vec{X}) \subseteq \sigma_k$. As a result, according to the MPCP problem to DBAT translation mechanism, it holds that $L(S_0) \prec\!\!\!- \in \mathcal{D}_{S_0}$. Therefore, $\mathcal{D}|\sim L(S_0)$.

(Induction step) We assume that the proposition holds for a plan of length $n$, and we show that it holds for a plan of length $n+1$. To do so, we need to show that if $L \subseteq \sigma_{n+1}$, where $\sigma_{n+1} = \gamma(\langle A_1, \ldots, A_{n+1}\rangle, I)$, then it holds that $\mathcal{D}|\sim L(S_{n+1})$, where $S_{n+1} = do([A_1, \ldots, A_{n+1}], S_0)$. $L \subseteq \sigma_{n+1}$ may be the case either due to an effect of action $A_{n+1}$ or inertia.

1. If $L$ is added in $\sigma_{n+1}$ as an effect of an action, then there exists an agent $i \in N$ with $pre_i(A_{n+1}) \subseteq \sigma_n$, and there exists an effect $\langle C, L\rangle \in eff_i(A_{n+1})$ such that $C \subseteq \sigma_n$, where state $\sigma_n = \gamma(\langle A_1, \ldots, A_n\rangle, I)$.

From the assumption of the induction step, it holds that for every $C \in pre_i(A_{n+1}) \cup \mathcal{C}$ it is the case that $\mathcal{D} \mid\sim C(S_n)$, where $S_n = do([A_1, \ldots, A_n], S_0)$.

From the translation mechanism and the grounding process we know that $ground(\mathcal{D}, S_{n+1})$ contains the following axiom:

$$L(S_{n+1}) \prec\!\!- \bigwedge_{C \in pre_i(A_{n+1}) \cup \mathcal{C}} C[S_n] .$$

Therefore, as there exists a defeasible derivation for every condition in the body of the rule, $\mathcal{D} \mid\sim L(S_{n+1})$.

2. If $L$ is part of $\sigma_{n+1}$ due to inertia, then it must be the case that $L$ is also part of $\sigma_n$.

   From the assumption of the induction step, we derive that:

   $$L \in \sigma_n \Rightarrow \mathcal{D} \mid\sim L(S_n) .$$

   From the translation mechanism of MPCP problems into DBATs and the grounding process, we know that $ground(\mathcal{D}, S_{n+1})$ contains a frame axiom of the following form (Observation 1):

   $$L(S_{n+1}) \prec\!\!- L(S_n), not\, C_1(S_n), \ldots, not\, C_m(S_n) .$$

   In the special case in which there is no specification of $A_{n+1}$ which contains an effect producing $\overline{L}$, the ground defeasible frame axiom takes the form:

   $$L(S_{n+1}) \prec\!\!- L(S_n) .$$

   In any case, since $\mathcal{D} \mid\sim L(S)$ and if any other literals appear in the body of the rule are assumptions, we conclude that $\mathcal{D} \mid\sim L(S_{n+1})$.

□

In order to ensure the correctness of the translation mechanism, defeasible derivations made from DBATs with complete initial states, but which cannot be derived from the corresponding MPCP problem, must not be warranted. To show that this is the case, we need the following proposition.

**Proposition 11.** *Let a MPCP problem $P = \langle N, F, I, O, G \rangle$, and the corresponding DBAT $\mathcal{D} = \langle \mathcal{D}_{ss}, \mathcal{D}_{ap}, \mathcal{D}_{S_0}, \mathcal{D}_{una}, \mathcal{D}_c \rangle$, both with complete initial states. Also, let $\sigma_k = \gamma(\langle A_1, \ldots, A_k \rangle, I)$ the state reached after performing the sequence of actions $\langle A_1, \ldots, A_k \rangle$, and the corresponding situation $S_k = do([A_1, \ldots, A_k], S_0)$. For every ground predicate literal L, if $L \in \sigma_k$ and $\overline{L} \notin \sigma_k$, then it holds that $\mathcal{D} |\!\approx L(S_k)$.*

*Proof.* Proof by induction on the length of the plan $k$.

(Base Case) $k = 0$. From the hypothesis, we have $L \in \sigma_0$ and $\overline{L} \notin \sigma_0$. From the translation mechanism of MPCP problems to DBATs, we derive that $L(S_0) \rightarrowtail \in \mathcal{D}$, $\overline{L}(S_0) \rightarrowtail \notin \mathcal{D}$, and that there is no other defeasible rule in $\mathcal{D}$ with head $L(S_0)$ or $\overline{L}(S_0)$. As a result, the argument $\langle \{L(S_0) \rightarrowtail \in \mathcal{D}\}, L(S_0) \rangle$ can be constructed from any ground instance of the theory, and there exists no defeater for this argument. Therefore, $\mathcal{D} |\!\approx L(S_0)$.

(Induction Step) We assume that the proposition holds for $k = n$. So, for every ground literal $L$, if $L \in \sigma_n$ and $\overline{L} \notin \sigma_n$, then it holds that $\mathcal{D} |\!\approx L(S_n)$. Next, we show that the proposition holds for $k = n+1$. To do so we first explain that $L \in \sigma_{n+1}$ and $\overline{L} \notin \sigma_{n+1} \Rightarrow \mathcal{D} |\!\not\approx \overline{L}(S_{n+1})$ by proving that every argument $\alpha$ claiming $\overline{L}(S_{n+1})$ is defeated. Finally, we show that $L \in \sigma_{n+1}$ and $\overline{L} \notin \sigma_{n+1} \Rightarrow \mathcal{D} |\!\approx L(S_{n+1})$ by differentiating between the two cases: $\overline{L} \in \sigma_n$ and $\overline{L} \notin \sigma_n$.

Proof of induction step:

1 Show that $\mathcal{D} |\!\not\approx \overline{L}(S_{n+1})$ by proving that every argument claiming $\overline{L}(S_{n+1})$ is defeated by an acceptable argument which attacks its supporting conditions (and not its claim). To do so we distinguish between the cases in which the final derivation step is made using an effect axiom and a frame axiom.

1a Consider the case in which there exists an argument claiming $\overline{L}(S_{n+1})$, in which the final derivation step is made using an effect axiom.

$\overline{L} \notin \sigma_{n+1}$ entails that in every agent specification there is no effect producing $\overline{L}$ that is applicable in $\sigma_n$. Formally:

$$\forall i \in N, \forall \langle \mathcal{C}_{\overline{L}}, \overline{L} \rangle \in \mathit{eff}_i \text{ it holds that } \exists C_{\overline{L}} \in \mathcal{C}_{\overline{L}} \cup \mathit{pre}_i \text{ such that } C_{\overline{L}} \notin \sigma_n .$$

From the state completeness assumption and Proposition 8, we have that:

$$C_{\overline{L}} \notin \sigma_n \Rightarrow \overline{C_{\overline{L}}} \in \sigma_n .$$

From the assumption of the induction step, we derive that:

$$C_{\overline{L}} \notin \sigma_n \text{ and } \overline{C_{\overline{L}}} \in \sigma_n \Rightarrow \mathcal{D} \mid\!\approx \overline{C_{\overline{L}}}.$$

Using Proposition 7 we infer that

$$\mathcal{D} \mid\!\approx \overline{C_{\overline{L}}} \Rightarrow \mathcal{D} \mid\!\not\approx C_{\overline{L}}.$$

From the MPCP problem to BAT translation mechanism and the grounding process, we know that ground effect axioms have the form:

$$\overline{L}(S_{n+1}) \prec \bigwedge_{C_{\overline{L}} \in \mathcal{C}_{\overline{L}} \cup pre_i(A_{n+1})} C_{\overline{L}}[S_n]\,,$$

where $i \in N$ is an agent and $\mathcal{C}_{\overline{L}}$ is the set of conditions for an effect $\langle \mathcal{C}_{\overline{L}}, \overline{L} \rangle \in eff_i(A_{n+1})$.

For every effect axiom with head $\overline{L}(S_{n+1})$, there exists at least one literal $C_{\overline{L}}$ in its body that is attacked and not warranted, since its complement is warranted.

As a result, every argument claiming $\overline{L}(S_{n+1})$, that uses an effect axiom for the final derivation step, is defeated by an acceptable argument attacking the conditions of the final derivation step.

1b Consider the case in which there exists an argument claiming $\overline{L}(S_{n+1})$, in which the final derivation step is made using a frame axiom. There are two cases: $\overline{L} \notin \sigma_n$ and $\overline{L} \in \sigma_n$.

- If $\overline{L} \notin \sigma_n$, from the state completeness assumption and Proposition 8 we infer that:

  $$\overline{L} \notin \sigma_n \Rightarrow L \in \sigma_n\,.$$

  From the assumption of the induction step we infer that:

  $$\overline{L} \notin \sigma_n \text{ and } L \in \sigma_n \Rightarrow \mathcal{D} \mid\!\approx L(S_n)\,.$$

  Therefore, if $\overline{L} \notin \sigma_n$, then every argument which claims $\overline{L}(S_{n+1})$ and uses a frame axiom for the final derivation step is defeated by an attack in its condition $\overline{L}(S_n)$ by an acceptable argument claiming $L(S_n)$.
- If $\overline{L} \in \sigma_n$, and since $\overline{L} \notin \sigma_{n+1}$, we reach the conclusion that every agent's specification has an effect producing $L$ which is applicable in

$\sigma_n$ without any ambiguity. Formally:

$$\forall i \in N, \text{ there exists } \langle C, L\rangle \in \mathit{eff}_i \text{ such that } \forall C \in \mathcal{C} \cup \mathit{pre}_i \text{ it holds that } C \in \sigma_n \text{ and } \overline{C} \notin \sigma_n .$$

From the assumption of the induction step we have that:

$$C \in \sigma_n \text{ and } \overline{C} \notin \sigma_n \Rightarrow \mathcal{D} |\!\approx C(S_n) .$$

Every ground frame axiom for $\overline{L}(S_{n+1})$ has the form:

$$\overline{L}(S_{n+1}) \multimap \overline{L}(S_n), not\, C_1(S_n), \ldots, not\, C_m(S_n) .$$

A ground frame axiom with head $\overline{L}$ is constructed for every possible combination of one condition (including preconditions) per effect producing $L$ (Observation 1). Therefore, every frame axiom with head $\overline{L}(S_{n+1})$ in $\mathit{ground}(\mathcal{D}, S_{n+1})$ contains one condition $C$ in its body, such that $C \in \mathcal{C} \cup \mathit{pre}_i(A_{n+1})$ . Since, for every such condition $C$ it holds that $\mathcal{D} |\!\approx C(S_n)$, every argument that uses the frame axiom for the derivation of its conclusions is defeated by an acceptable argument attacking one of its assumptions.

From (1a) and (1b), we infer that if there exists an argument claiming $\overline{L}(S_{n+1})$, then this argument is defeated by an acceptable argument attacking its supporting conditions (and not its claim).

2 Show that $\mathcal{D} |\!\approx L(S_{n+1})$, by distinguishing between the following cases: $\overline{L} \in \sigma_n$ and $\overline{L} \notin \sigma_n$.

2a If $\overline{L} \in \sigma_n$, and since we know from the hypothesis that $\overline{L} \notin \sigma_{n+1}$, we derive that every agent's specification contains an effect producing $L$ that is applicable in $\sigma_n$ without any ambiguity. Formally:

$$\forall i \in N \text{ there exists } \langle C, L\rangle \in \mathit{eff}_i(A_{n+1}) \text{ such that } \forall C \in \mathcal{C} \cup \mathit{pre}_i(A_{n+1}) \text{ it holds that } C \in \sigma_n \text{ and } \overline{C} \notin \sigma_n .$$

From the assumption of the induction step, we have that $\forall C \in \mathcal{C} \cup \mathit{pre}_i(A_{n+1})$:

$$C \in \sigma_n \text{ and } \overline{C} \notin \sigma_n \Rightarrow \mathcal{D} |\!\approx C(S_n) .$$

From the translation mechanism from MPCP problems to DBATs, and the grounding process, we know that for any $i \in N$ that there exists an axiom in $ground(D, S_{n+1})$ such that:

$$L(S_{n+1}) \prec \bigwedge_{C \in \mathcal{C} \cup pre_i(A_{n+1})} C[S_n],$$

where $\langle \mathcal{C}, L \rangle \in eff_i(A_{n+1})$, and every literal $C(S_n)$ in the body of the rule is warranted.

Since $L \in \sigma_{n+1}$, there exists a defeasible derivation for $L(S_{n+1})$ (Proposition 10).

As a result, there exists an argument $\alpha$ claiming $L(S_{n+1})$, whose final derivation step is done using an effect axiom, the conditions of which are all warranted. Also, according to (1), every argument $\beta$ claiming $\overline{L}(S_{n+1})$ if defeated by an acceptable argument $\alpha'$ different from $\alpha$, since the defeater does not attack the claim of $\beta$. Therefore, $\alpha$ is defended against every defeat by acceptable arguments different from itself. As a result, if $\overline{L} \in \sigma_n$, $\mathcal{D} |\!\approx L(S_{n+1})$.

2b If $\overline{L} \notin \sigma_n$, from the completeness assumption we derive that:

$$L \in \sigma_n .$$

From the induction step's assumption we have that:

$$\overline{L} \notin \sigma_n \text{ and } L \in \sigma_n \Rightarrow \mathcal{D} |\!\approx L(S_n) .$$

From (1a) we know that for every effect axiom with head $\overline{L}(S_{n+1})$ there exists at least one literal $C_{\overline{L}}$ in its body that is attacked and not warranted. As a result, every argument claiming $\overline{L}(S_{n+1})$, that uses an effect axiom for the final derivation step, is defeated.

From the MPCP problem to DBAT translation mechanism, and the grounding mechanism, we know that there exists a frame axiom with the form:

$$L(S_{n+1}) \prec L(S_n), not\, C^1_{\overline{L}}, \ldots, not\, C^m_{\overline{L}},$$

whose body contains the default negated literals that are attacked and not warranted. This is the case, since every frame axiom contains one literal for each specification and conditional effect that produces the complement

of the literal in the head of the axiom (Observation 1). Also, since a frame axiom is produced for every possible combination of conditions, there exists a ground frame axiom in $ground(\mathcal{D}, S_{n+1})$ whose every assumption can be defended against every possible attack.

Therefore, $\alpha$ is defended against every defeat by an acceptable argument different than itself. As a result, there exists an acceptable argument claiming $L(S_{n+1})$, and $\mathcal{D} |\approx L(S_{n+1})$.

From (2a) and (2b) we conclude that: $\mathcal{D} |\approx L(S_{n+1})$.

□

Using these results we conclude that if a ground literal with a ground situation term $S$ is warranted from a DBAT with a corresponding MPCP problem $P$, then we can derive from $P$ that this literal is part of the state that corresponds to $S$.

**Proposition 12.** *Let a MPCP problem $P = \langle N, F, I, O, G \rangle$, and the corresponding DBAT $\mathcal{D} = \langle \mathcal{D}_{ss}, \mathcal{D}_{ap}, \mathcal{D}_{S_0}, \mathcal{D}_{una}, \mathcal{D}_c \rangle$, both with complete initial states. Also, let the state $\sigma_k = \gamma(\langle A_1, \ldots, A_k \rangle, I)$ reached after performing the sequence of actions $\langle A_1, \ldots, A_k \rangle$, and the corresponding situation $S_k = do([A_1, \ldots, A_k], S_0)$. For every ground predicate literal L, if $\mathcal{D} |\approx L(S_k)$ then $L \in \sigma_k$.*

*Proof.* Proof by contradiction.

We assume that (1) $\mathcal{D} |\approx L(S_k)$ and (2) $L \notin \sigma_k$ hold.

The initial state is complete. So from Proposition 8 we infer that $\overline{L} \in \sigma_k$ (3).

From (2), (3), and Proposition 11, it follows that $\mathcal{D} |\approx \overline{L}(S_k)$ (4).

From (4) and Proposition 7, we infer that $\mathcal{D} |\not\approx L(S_k)$ (5).

(5) directly contradicts hypothesis (1). As a result, the proposition holds. □

Using the above propositions we show that, for every DBAT $\mathcal{D}$ that corresponds to a MPCP problem $P$, every plan that is warranted from $\mathcal{D}$ is a candidate plan for $P$. This result illustrates the correctness of the translation mechanism.

**Proposition 13.** *Let a MPCP problem $P = \langle N, F, I, O, G \rangle$, and the corresponding DBAT $\mathcal{D} = \langle \mathcal{D}_{ss}, \mathcal{D}_{ap}, \mathcal{D}_{S_0}, \mathcal{D}_{una}, \mathcal{D}_c \rangle$, both with complete initial states. Also, let the state $\sigma_k = \gamma(\langle A_1, \ldots, A_k \rangle, I)$ reached after performing the sequence of actions $\pi = \langle A_1, \ldots, A_k \rangle$, and the corresponding situation $S_j = do([A_1, \ldots, A_j], S_0)$, for $j = 0, \ldots, k$ . If $\mathcal{D} |\approx Poss(A_1, S_0), Poss(A_2, S_1), \ldots, Poss(A_k, S_{k-1}), G_1(S_k), G_2(S_k), \ldots, G_m(S_k)$, then $\pi$ is a candidate plan for P.*

*Proof.* Proof by induction on the length of the plan $k$.
(Base Case) If $k = 0$ then $\mathcal{D} \mid\!\approx G_1(S_0), G_2(S_0), \ldots, G_m(S_0)$. Therefore,

$$\mathcal{D} \mid\!\approx G_1(S_0) \text{ and } \mathcal{D} \mid\!\approx G_2(S_0) \text{ and } \ldots \text{ and } \mathcal{D} \mid\!\approx G_m(S_0)\,.$$

From Proposition 12 we have, $G_1, G_2, \ldots, G_m \subseteq I$. As a result, the empty plan is a candidate plan for $P$.

(Induction Step) We assume that the proposition holds for every plan of length $k = n$. We show that it also holds for plans of length $k = n+1$.

From the hypothesis, we have that $\mathcal{D} \mid\!\approx Poss(A_1, S_0), Poss(A_2, S_1), \ldots, Poss(A_n, S_{n-1}), Poss(A_{n+1}, S_n), G_1(S_{n+1}), G_2(S_{n+1}), \ldots, G_m(S_{n+1})$. Or equivalently:

$$\mathcal{D} \mid\!\approx Poss(A_1, S_0), Poss(A_2, S_1), \ldots, Poss(A_n, S_{n-1})\,.$$

So, since $\langle A_1, \ldots, A_n \rangle$ is a warranted plan from $\mathcal{D}$ with respect to the empty goal, according to the induction step, it is also a candidate plan.

Also, $\mathcal{D} \mid\!\approx Poss(A_{n+1}, S_n)$. Therefore, according to the translation mechanism of MPCP problems to DBATs, there exists an agent $i$ such that $\mathcal{D} \mid\!\approx \bigwedge_{C \in pre_i(A_{n+1})}$. As a result, $\forall C \in pre_i(A_{n+1}), \mathcal{D} \mid\!\approx C$, and according to Proposition 12:

$$\forall C \in pre_i(A_{n+1}), C \in \sigma_n\,.$$

As a result, there exists at least one specification in which the action $A_{n+1}$ is applicable. Therefore, the plan $\langle A_1, \ldots, A_{n+1} \rangle$ is applicable in $I$ (1).

$$\mathcal{D} \mid\!\approx G_1(S_{n+1}) \text{ and } \mathcal{D} \mid\!\approx G_2(S_{n+1}) \text{ and } \ldots \text{ and } \mathcal{D} \mid\!\approx G_m(S_{n+1})\,.$$

As a result, from Proposition 12 we have:

$$G_1, G_2, \ldots, G_m \subseteq \sigma_{n+1} \text{ (2).}$$

From (1) and (2), we derive that the plan $\langle A_1, \ldots, A_{n+1} \rangle$ is a candidate plan. □

## 3.5 Extensions

DBATs subsume MPCP problems with respect to the expressive power of the formalisms. This section investigates the expressiveness of the defeasible situation calculus formalism, and look into extensions to defeasible basic action theories which provide additional expressive power that our argumentation-based formalism handles without modifications.

### 3.5.1 Ramifications

This section describes extended defeasible action theories including rules that represent axiomatic beliefs about the domain that go beyond what is considered to be classical planning knowledge. We describe how axioms like ramifications and domain constraints can be represented with the language and describe how the argumentation-based reasoning mechanism handles the reasoning overhead.

The argumentation mechanism can deal with rules that do not coincide with the structure of the defeasible basic action theory axioms. For instance, such rules are ramifications or state constraints as for example a rule stating that if an object is in one position it cannot be in a different position in the same situation. In general, by *extended defeasible action theories* we consider defeasible basic action theories with domain constraints.

Our framework treats ramifications just like every other axiom. Contradictions among the different axioms are resolved by the argumentation mechanism. The system does not need to differentiate between the different axioms. This enables a uniform treatment of reasoning steps made with different types of axioms.

Even though the reasoning mechanism is able to deal with such rules, reasoning becomes more demanding. Contrary to derivations with defeasible basic action theories, every derivation step does not lead from a situation to its predecessor, as there may be multiple steps regarding fluents of the same situation. Cycle detection is necessary in the search for derivation, since such arguments may lead to circles in inferences.

**Definition 26.** *An* extended defeasible action theory *is a tuple*

$$\mathcal{D} = \langle \mathcal{D}_{ss}, \mathcal{D}_{ap}, \mathcal{D}_{dc}, \mathcal{D}_{una}, \mathcal{D}_{S_0} \rangle$$

*containing defeasible rules describing defeasible successor state axioms, action preconditions of actions, domain constraints, presumptions regarding the initial situation and non-fluent beliefs and unique names axioms for actions.*

*Defeasible domain constraints* detail rules describing relations between predicates in the same situation. The head of a defeasible domain constraint is a literal predicate regarding a situation $s$ (e.g. $F(\vec{x},s)$ or $\sim F(\vec{x},s)$), and the body of the rule may be any disjunction of conjunctions of literal predicates. Only one situation term is allowed to appear in such rules.

**Example 3.** *An example of a defeasible domain constraint is the belief that if there is evidence that an object is in a position, there are reasons to believe that it is not in a different position:*

$$\sim At(x,l,s) \multimap At(x,l',s), l \neq l'.$$

*This domain constraint implicitly represents the belief that it is not possible for an object to be in two different positions in the same situation.*

Domain constraints can be used to identify errors in the reasoning process. Consider for instance the domain constraint $\sim At(x,l,s) \multimap At(x,l',s), l \neq l'$. Using this constraint we can attack arguments which are supported by beliefs that do not coincide with the knowledge incorporated within the domain constraints.

Additionally, domain constraints can represent ramifications or indirect effects, i.e. side-effects of actions. Consider for example the belief that switching on the lamp produces light, and light always produces heat. This enables a more straightforward representation of the domain differentiating between the direct effects of actions and their side-effects, which may be potentially insignificant.

Reasoning is performed in exactly the same way as in defeasible basic action theories. The agent first needs to come up with a derivation, providing the reasons to believe the desired claim, and then the argumentation process needs to identify whether the corresponding argument is acceptable. Derivations in extended defeasible action theories may involve multiple inference steps regarding fluents about the same situation. This is the main difference with derivations made from defeasible basic action theories.

### 3.5.2 Observations

Consider a situation in which the agents agree on a plan and start executing it until an action cannot be performed. In order to cope with the plan execution failure they must re-plan while taking into account all the monitoring information they have

distributively acquired. In complex, partially observable domains agents may monitor different aspects of the system, and their collective observations may still offer a limited view of the current state of the environment.

The collective beliefs of the agents include:

- Initial situation view of the environment
- Operator specifications
- Sequence of actions executed
- Observations from the traversed states

These beliefs can be used within a slightly different setup in order to synthesise an alternative plan without progressing their entire theory to the current state. Reasoning is performed based on their initial theory $\mathcal{D} = \langle \mathcal{D}_{ss}, \mathcal{D}_{ap}, \mathcal{D}_{S_0}, \mathcal{D}_{una}, \mathcal{D}_c \rangle$, the set $\mathcal{D}_o$ and the sequence of executed actions $\pi_c$ and the relevant current situation $S_c = do([\pi], S_0)$. $\mathcal{D}_o$ contains observations of the form $L(S_k) \prec$, where $L$ is a ground fluent predicate, with its situation term $S_k$ being a predecessor situation of $S_c$ and a successor situation to $S_0$.

**Definition 27.** *Suppose theory $\mathcal{D}' = \mathcal{D} \cup \mathcal{D}_o$ comprising of a defeasible basic action theory and a set of observations after the execution of the sequence of actions $\pi_c$ , and let $AF = \langle Args, Defs \rangle$ be the argumentation framework for all arguments that can be constructed from $\mathcal{D}'$, and $GE_{AF}$ its grounded extension. Consider the shared goals of the agents $G_1, \ldots, G_m$. The sequence of actions $\pi = \langle A_1, A_2, \ldots, A_n \rangle$ is a warranted plan if and only if:*

$$\mathcal{D} \mid\!\approx Poss(A_1, S_c), Poss(A_2, S_{c+1}), \ldots, Poss(A_n, S_{c+n-1}), G_1(S_{c+n}), \ldots, G_m(S_{c+n}),$$

*where $S_c = do([\pi_c], S_0)$ is the current situation, $S_i = do(A_i, S_{i-1})$ for $i > c$ denotes the situation resulting from the application of action $A_i$ to the predecessor situation $S_{i-1}$.*

The defeat relation reflect that recent observations are more accurate. This can be done by simply assigning to recent observations higher preference values than outdated planning beliefs. As a result, arguments made using recent observations will be preferred over arguments that combine outdated information with beliefs from the operator specifications to predict the outcome of the agents' actions.

### 3.5.3 Default Negated Conditions

Default negation allows the representation of two types of conditions (both preconditions and conditional effects): conditions which must be known to hold, and conditions for which it is sufficient to have no evidence that their complement is the case. The first is the usual case. Conditions must be defeasibly derived and defended against all counterarguments. The latter can be used to specify special cases, in environments with high degrees of uncertainty, allowing the agent to reach conclusions quickly without considering less significant conditions, or special cases which rarely hold and in most cases agents are unaware about their status. These can be investigated during the argumentation stage in the face of sufficient evidence.

Consider the following rule, stating that there is light in the room after we switch on the light provided that there is electricity in the building:

$$Light(do(switch\_on, s)) \prec Electricity(s)\,.$$

If we have no evidence related to whether the predicate $Electricity(S)$ (for a ground situation term $S$) has a positive or a negative value, we do not have reasons to believe that $Light(do(switch\_on, S))$ holds. Consider a domain in which lack of electricity is an unusual special case. We can represent this knowledge by modifying the rule using default negation:

$$Light(do(switch\_on, s)) \prec Light(s), not \sim Electricity(s)\,.$$

This modification asserts that the rule is applicable regardless of whether there are reasons to believe $Electricity(S)$ or $\sim Electricity(S)$. However, the conclusions that are derived using this rule are not warranted if there exists strong evidence towards $\sim Electricity(S)$. Therefore, the difference between the two rules lies in the case that $Electricity(S)$ cannot be defeasibly derived from the agents' theory, and $\sim Electricity(S)$ is not warranted. The first rule results in $Light(do(switch\_on, S))$ not being defeasibly derivable, whereas it is derivable in the second, and warranted provided that $\sim Light(do(switch\_on, S))$ is not warranted.

Assuming that this is the only action affecting the value of the predicate *Light*, we also have the following axiom for the negative literal:

$$\sim Light(do(a, s)) \prec \sim Light(s), a \neq switch\_on; \sim Light(s), \sim Electricity(s)\,.$$

In this case, the condition $\sim Electricity(s)$ is preceded by strong negation, since the case in which there is no electricity in the building is believed to be unusual, and the agents should assume it does not hold, unless there is strong evidence to the contrary.

### 3.5.4 Partially Warranted Plans

According to our definition of plans, all actions should be executable in sequence and their application should result in states that achieve the shared goal. In domains of great uncertainty, the agents may not be able to find a plan, if there is not enough information to account for all the necessary conditions. Our aforementioned definition of an acceptable plan is not useful in this case.

We provide a relaxed solution concept based on default negation. The idea is that if the agents cannot synthesise a plan whose every action is believed to be executable, they can fall back on a search for plans which contain actions that are not believed to be inapplicable.

Partially warranted plans are defined as follows:

**Definition 28.** *Suppose a defeasible action theory $\mathcal{D}$, and let the argumentation framework $AF = \langle Args, Defs \rangle$ including all arguments that can be constructed from $\mathcal{D}$, and its grounded extension $GE_{AF}$. Consider the shared goals of the agents $G_1, \ldots, G_m$. The sequence of actions $\pi = \langle A_1, A_2, \ldots, A_n \rangle$ is a partially warranted plan iff*

$$\mathcal{D} \mid\!\approx not \sim Poss(A_1, S_0), \ldots, not \sim Poss(A_n, S_{n-1}), G_1(S_n), \ldots, G_m(S_n),$$

*where $S_i = do(A_i, S_{i-1})$ denotes the situation resulting from the application of action $A_i$ to the predecessor situation $S_{i-1}$.*

The concept of partially warranted plans is strictly more general than the concept of warranted plans.

**Proposition 14.** *Suppose a defeasible action theory $\mathcal{D}$, and let $AF = \langle Args, Defs \rangle$ the argumentation framework for all arguments that can be constructed from $\mathcal{D}$, and $GE_{AF}$ its grounded extension. Consider $G_1, \ldots, G_m$ to be the shared goals of the agents. If $\pi = \langle A_1, A_2, \ldots, A_n \rangle$ is a warranted plan, then $\pi = \langle A_1, A_2, \ldots, A_n \rangle$ is a partially warranted plan.*

*Proof.* In order for the plan to be partially warranted every (default negated) literal in the sequence must be warranted. For every goal literal $G_i$, since the plan is warranted it holds that $\mathcal{D} \mid\!\approx G_i(S_n)$, where $S_n = do([A_1, \ldots, A_n], S_0)$. We need to show that for every action of the plan $A_k$ and every situation $S_{k-1} = do([A_1, \ldots, A_{k-2}], S_0)$ it holds that $\mathcal{D} \mid\!\approx not \sim Poss(A_k, S_{k-1})$. Since the plan is warranted, $\mathcal{D} \mid\!\approx Poss(A_k, S_{k-1})$. Therefore, every argument claiming $not\, Poss(A_k, S_{k-1})$ is defeated. From the specification of default negation, $\mathcal{D} \mid\sim not \sim Poss(A_k, S_{k-1})$. Also, every argument attacking the assumption $not \sim Poss(A_k, S_{k-1})$ is defeated. As a result, $\mathcal{D} \mid\!\approx not \sim Poss(A_k, S_{k-1})$. □

The contrary does not hold. Consider a plan with a single action and a single precondition, for which we do not have neither evidence for or evidence against. If we can infer that this action achieves the goal, we can infer that the plan is partially warranted. However, since we do not have any indication that the action's precondition holds in the initial situation, we cannot construct an argument for the plan, and the plan is not warranted.

Potential weakly acceptable plans may be ordered by the "degree of uncertainty" in the applicability of their actions (or conditions which are not supported by any evidence). This is a measure of the assumptions that cannot be defeated, but at the same time are not warranted.

## 3.6 Summary

Multi-perspective cooperative planning deals with the problem of synthesising plans in domains where agents have contradictory views regarding the initial state of the environment and operator specifications. Based on the agents' planning domain beliefs, and using standard, set-theoretic planning notation we define the multi-perspective cooperative planning problem, and specify the solution concept of candidate plans. Candidate plans are rather weak, since they do not account for potential objections to these plans. Defeasible situation calculus enables the encoding of MPCP in the form of defeasible basic action theories, and formalises the notion of plan acceptability. This allows the specification of the warranted plan solution concept, which asserts that the plan can be defended against every possible objection. Defeasible situation calculus subsumes MPCP problems in terms of expressive power. In addition, we provide a translation mechanism which bridges the two formalisms and enables the use of modern planning techniques to solve MPCP problems.

The main contributions of this chapter are:

- Formalisation of the MPCP problem.
- Specification of an expressive language for reasoning about contradictory dynamic domains.
- Concretisation of the notion of acceptability in planning domains.
- Bridging automated planning, reasoning about action and argumentation.

# Chapter 4

# Reasoning and Planning Algorithms

## 4.1 Introduction

Chapter 3 defines an argumentation-based framework, which formalises the notion of plan acceptability in multi-perspective co-operative planning domains. This chapter focuses on the algorithmic problem of synthesising warranted plans, and is separated into two main parts. Initially, we focus on the defeasible situation calculus formalism and describe how planning can be performed using DBATs. We focus on inherent characteristics of the planning domain that can be exploited, in order to simplify the necessary tasks and improve the efficiency of the process. The second part of this chapter is based on the set-theoretic planning representation. We look at the problem from a "classical" planning perspective and explain how search for candidate plans can be delegated to efficient state-of-the-art planners. In this way, we exploit the planners' highly optimised, heuristic search, further improving the practicality of our approach.

## 4.2 Planning with DBATs

Following the definition of a warranted plan (Definition 24), planning with a DBAT $\mathcal{D}$ can be performed by searching for a ground situation $S$, such that:

$$\mathcal{D} \mid\!\approx executable(S), goals(S)\,.$$

$S$ represents the history of the execution of the plan's actions in sequence. The special predicate *goals* abbreviates the expression $G_1(S), G_2(S), \ldots, G_m(S)$, where literals $G_1, G_2, \ldots, G_m$ represent the agents' goal predicates. The special predicate $executable(S)$

is also an abbreviation and is defined for all ground situation terms $S$:

$$executable(S) \equiv \begin{cases} \text{true} & \text{if } S = S_0 \\ executable(S'), Poss(A, S') & \text{otherwise with } S = do(A, S') \end{cases}$$

For example, the predicate $executable(do(A_2, do(A_1, S_0)))$ abbreviates the expression $Poss(A_2, do(A_1, S_0)), Poss(A_1, S_0)$. The abbreviation $executable(S_0)$ holds by definition, and is omitted from the expression.

### 4.2.1 A Simple Exhaustive Planner

Algorithm 1 implements a simple exhaustive planner, which searches the plan space for a warranted plan. We restrict the search to plans of a reasonable size $\varepsilon$, by considering only situation terms in the set $\mathcal{S}_\varepsilon = \{S \mid S \text{ is a predecessor of a ground situation term of length } \varepsilon\}$. The exhaustive nature of the search for a warranted plan asserts that the planning process is *sound* and *complete* for problems with a solution of at most length $\varepsilon$.

**Algorithm 1:** An exhaustive planer based on propositional argumentation

$ground(\mathcal{D}) := \bigcup_{\forall S_\varepsilon \in \mathcal{S}_\varepsilon} ground(\mathcal{D}, S_\varepsilon)$;
$Args := \{\alpha \mid \alpha \text{ is an argument that can be constructed from } ground(\mathcal{D})\}$;
$Defs := \{(\alpha, \beta) \mid \alpha, \beta \in Args \text{ and } \alpha \text{ defeats } \beta\}$;
$AF := \langle Args, Defs \rangle$;
$Warranted := \{Claim(\alpha) \mid \alpha \text{ is acceptable w.r.t. } AF\}$;
**repeat**
  Select a new sequence $\pi := \langle A_1, A_2, \ldots, A_n \rangle$, with $n := [0, \varepsilon)$;
  **if** $Warranted \supseteq \{L(S_\pi) \mid L(S_\pi)$ *appears in* $executable(S_\pi)$ *or* $goals(S_\pi)\}$
  **then**
    **return** $\pi$;
**until** *there does not exist a new plan*;
**return** null;

Algorithm 1 is built on top of a reasoning component which is capable of evaluating the warrant state of ground queries of the form $\mathcal{D} \mid\!\approx L(S)$, where $L(S)$ is a ground literal with a ground situation term $S$. This process involves the following tasks:

- Theory grounding.

- Argument Generation.
- Calculation of defeats among arguments.
- Establishing argument acceptability (with respect to the employed argumentation semantics).
- Search for acceptable arguments claiming the literals appearing in the query.

Algorithm 1 conducts these tasks in a simple, straightforward manner. It evaluates whether a fluent is warranted by a defeasible basic action theory, and outlines the necessary steps that need to be undertaken to assert soundness and completeness. It is essentially a propositional argumentation mechanism reasoning over a "reasonably" maximal ground theory.

The grounding mechanism grounds $\mathcal{D}$ with respect to every situation referring to a history of less than $\varepsilon$ actions. The number $\varepsilon$ is used as a "reasonable" threshold. Argument generation and evaluation of argument acceptability are conducted in a simple propositional argumentation-based reasoning manner.

The generation of the argumentation framework involves generating all arguments and identifying the defeat relations among them. Argument generation can be performed by searching for defeasible derivations and discounting those that are based on a contradictory support. The defeat relations are identified by inspecting the arguments. Attacks are identified in the light of contradictory beliefs among arguments. Defeats are composed from attacks based on the employed defeat relation. This is an aspect of the framework that can be fine-tuned to incorporate meta-information about the domain. For instance, it may be the case that the more specific the conditions are in a conditional effect, the more credible the operator is.

The evaluation of argument acceptability is performed with respect to the employed argumentation semantics. For example, if grounded (sceptical) argumentation semantics are used, we must identify which arguments are part of the grounded extension of the argumentation framework. This task can be performed by a labelling process (Modgil and Caminada, 2009), which follows the employed argumentation semantics.

Algorithm 1 is useful as an outline of the basic tasks that are necessary for the synthesis of warranted plans. However, the use of general propositional argumentation methods to synthesise warranted plans is not efficient. The main problem here arises due to the overall size of the generated ground defeasible theory, which makes the argumentation process, which is by its nature exhaustive, very inefficient.

### 4.2.2 Strategies and Heuristics

In order to maximise the efficiency of our methods, we focus on pruning strategies and heuristics that allow the reduction of the overall space we need to search for, in terms of beliefs, arguments and potential plans. The rationale behind these strategies is based on the analytical results presented in Chapter 3.

#### 4.2.2.1 Situation-Dependent Grounding

Grounding the domain beliefs of the agents requires the substitution of the variables in the rules in the DBAT with constant terms, for every possible combination, while respecting the sorts of terms and the equalities and inequalities appearing in the bodies of the rules. This step is responsible to a great extent for the impracticality of Algorithm 1.

Even if we assume that the number of objects of the planning domain is manageable, we have to account for every situation term corresponding to a potential plan. Since situation terms represent sequences of actions, we use $\varepsilon$ to restrict the length of potential plans to a "reasonable" threshold. However, this solution is still impractical since the set of literals, grounded for every situation up to length $k$, is exponential in $k$.

Following Proposition 4 on page 70, we observe that the derivability of a ground literal $L(S)$ is independent of any rules which refer to situations that are successors to $S$. Similarly, defeasible rules referring to situations that are neither predecessors nor successors to $S$ are also irrelevant, since all these rules refer to different sets of ground predicates.

**Observation 2.** *Consider a DBAT $\mathcal{D}$ and a ground literal $L(S)$, where $S$ is a ground situation term. Assume a threshold $\varepsilon$ and the set $\mathcal{S}_\varepsilon = \{S \mid S$ is a ground situation term of length $\varepsilon\}$, such that $S$ is equal or a predecessor of a situation $S' \in \mathcal{S}_\varepsilon$.*

$$\left( \bigcup_{S_\varepsilon \in \mathcal{S}_\varepsilon} ground(\mathcal{D}, S_\varepsilon) \right) |\sim L(S) \text{ if and only if } ground(\mathcal{D}, S) |\sim L(S)\,.$$

According to the specification of the grounding mechanism, the set $ground(\mathcal{D}, S)$ contains all defeasible rules that have been grounded with respect to situation terms that are equal to or predecessor of $S$. As a result, all rules that are relevant to the derivation of $L(S)$ appear in $ground(\mathcal{D}, S) |\approx L(S)$.

Equivalently, we generalise Proposition 5 on page 76, and conclude that the warrant status of every ground literal $L(S)$ is independent of any rules referring to situations that are not equivalent to or that are predecessors of $S$.

**Observation 3.** *Let the DBAT $\mathcal{D}$ and a ground literal $L(S)$, where $S$ is a ground situation term. Assume a threshold $\varepsilon$ and the set $\mathcal{S}_\varepsilon = \{S \mid S$ is a ground situation term of length $\varepsilon\}$, such that $S$ is equal or a predecessor of a situation $S' \in \mathcal{S}_\varepsilon$.*

$$\left(\bigcup_{S_\varepsilon \in \mathcal{S}_\varepsilon} ground(\mathcal{D}, S_\varepsilon)\right) |\!\approx L(S) \text{ if and only if } ground(\mathcal{D}, S) |\!\approx L(S)\,.$$

The above observations reflect that any conclusions regarding $\mathcal{D} |\!\sim L(S)$ and $\mathcal{D} |\!\approx L(S)$ can be drawn from the ground theory $ground(\mathcal{D}, S)$, and are independent of the beliefs $\left(\bigcup_{S_\varepsilon \in \mathcal{S}_\varepsilon} ground(\mathcal{D}, S_\varepsilon)\right) \setminus ground(\mathcal{D}, S)$.

Accordingly, in order to answer any query of the form $\mathcal{D} |\!\sim L(S)$ or $\mathcal{D} |\!\approx L(S)$, we ground the DBAT only with respect to $S$ and its predecessor situations. Such queries represent the main task for reasoning and planning with defeasible basic action theories.

The exhaustive planner described above is not suitable to utilise the grounding strategy, since planning is performed as a simple search within the results of the argumentation process. In the following sections, we describe how the planning process can be adapted, so that answering queries of the form $\mathcal{D} |\!\sim L(S)$ and $\mathcal{D} |\!\approx L(S)$ is placed at the heart of the planning algorithm.

#### 4.2.2.2 Argument and Defeater Generation

The exhaustive planner described in Algorithm 1 requires carrying out the argumentation process prior to the search for a warranted plan. Even with a ground theory of a restricted size this is impractical, since the argumentation process is expensive, primarily due to its exhaustive nature. To this end, the following observation focuses on restricting the argumentation generation process to arguments that are relevant to the query that is under evaluation.

**Observation 4.** *The warrant state of a ground literal $L(S)$ depends exclusively on the arguments with claim $L(S)$, their defeaters, the defeaters of their defeaters, etc.*

Formally, the set $Args_{L(S)} \subseteq Args$ containing all the arguments that are *relevant* to the warrant state of $L(S)$ is specified as follows:

$$Args_{L(S)} = \{\alpha \mid Claim(\alpha) = L(S) \text{ or } \exists \beta \in Args_{L(S)} \text{ such that } (\alpha, \beta) \in Defs\}\,.$$

$Args_{L(S)}$ contains arguments with claim $L(S)$, their defeaters, the defeaters of their defeaters, and so forth. Any argument in the set $Args \setminus Args_{L(S)}$ is irrelevant to the

**Algorithm 2:** Construct the set $Args_{L(S)}$, with respect to the query $\mathcal{D} |\!\approx L(S)$

$i := 0$;
$Args_{L(S)} := \emptyset$;
$Args^0_{L(S)} :=$ Generate all arguments with claim $L(S)$;
**while** $Args^i_{L(S)} \neq \emptyset$ **do**
  $Args^{i+i}_{L(S)} :=$ Generate all defeaters for arguments in $Args^i_{L(S)}$ ;
  $Args^{i+i}_{L(S)} := Args^{i+1}_{L(S)} \setminus Args_{L(S)}$ ;
  $Args_{L(S)} := Args_{L(S)} \cup Args^{i+i}_{L(S)}$ ;
**Return** $Args_{L(S)}$;

decision of whether the ground fluent $L(S)$ is warranted. As a result, constructing these arguments is unnecessary.

Additionally, every ground rule that does not appear within an argument in $Args_{L(S)}$ is also irrelevant to the query. Let $ground(\mathcal{D})_{Args_{L(S)}} \subseteq ground(\mathcal{D}, S_\varepsilon)$ be the subset of the ground theory containing only beliefs relevant to arguments in $Args_{L(S)}$:

$$ground(\mathcal{D})_{Args_{L(S)}} = \{\phi \mid \exists \alpha \in Args_{L(S)} \text{ such that } \phi \in \{Claim(\alpha)\} \cup Support(\alpha)\} .$$

Ground beliefs in the set $ground(\mathcal{D}, S_\varepsilon) \setminus ground(\mathcal{D})_{Args_{L(S)}}$ are not used in arguments that are relevant to the acceptability of $L(S)$.

In the worst case, if the warrant state of every literal in the theory depends on the values of all other literals in the theory, the sets described above would be equivalent to the general sets. This is rarely the case, especially in complex problems for large planning domains which usually include multiple beliefs that are irrelevant to the overall goal.

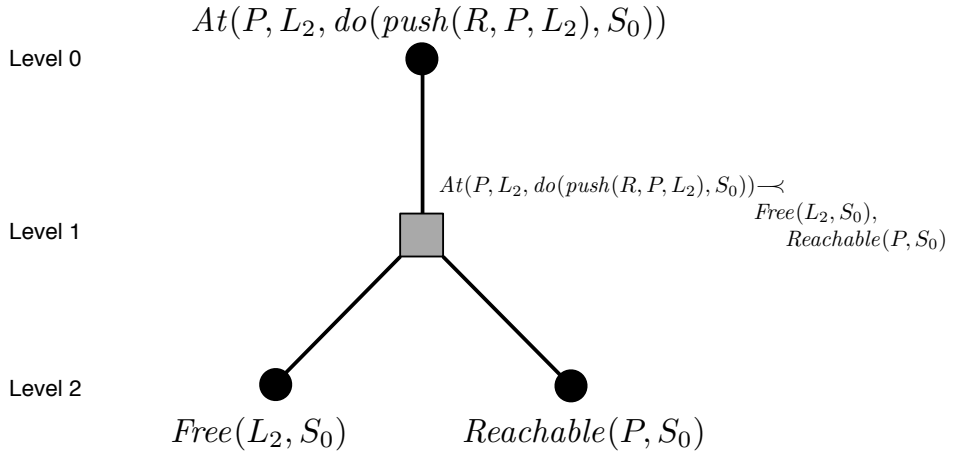
The complete argumentation framework can be viewed as a directed graph called an *argumentation graph*. Nodes correspond to the arguments of the argumentation framework, whereas the edges represents the defeats between them. Algorithm 2 constructs subgraphs of the argumentation graph. Every subgraph contains an argument $\alpha$ with $Claim(\alpha) = L(S)$, as well as every argument $\beta$, such that there is a path from the node corresponding to $\beta$ to the node representing $\alpha$. Every argument $\beta'$, such that there is no path from its corresponding node to any node representing an argument in $Args \setminus Args_{L(S)}$, are irrelevant to $L(S)$. Therefore, these are not included in the subgraph. In the worst case, there is a path from every node in the graph to an argument claiming $L(S)$, and as a result $Args_{L(S)} = Args$.

Algorithm 2 asserts that arguments that are irrelevant to a query are not generated and evaluated. This process is further optimised in the following chapter, in which we present a dialogue based approach which evaluates the acceptability of a plan through a dispute. The benefit of this process is that it does not always require the generation of all arguments. Algorithm 2 generates the relevant argument trees in an exhaustive breadth-first order, and labelling is performed after the tree has been constructed. This process is further optimised by performing labelling after every iteration.

**Example 4.** *Consider the processes of argument and defeater generation for the query* $At(P,L_2,do(push(R,P,L_2),S_0))$ *which denotes the belief that a parcel P is in a location* $L_2$ *after being pushed to this location. Initially, we generate arguments whose claim is the literal* $At(P,L_2,do(push(R,P,L_2),S_0))$*. For example, assume that the following argument is available.*

$$\alpha = \langle \{At(P,L_2,do(push(R,P,L_2),S_0)) \prec Free(L_2,S_0), Reachable(P,S_0), \\ Free(L_2,S_0) \prec, \ Reachable(P,S_0) \prec\}, \\ At(P,L_2,do(push(R,P,L_2),S_0))\rangle$$

*This argument states that the claim holds as a direct result of the action* $push(R,P,L_2)$*, because there are reasons to believe that the relevant conditions* $Free(L_2,S_0)$ *and* $Reachable(P,S_0)$ *are the case. This argument is depicted as follows:*



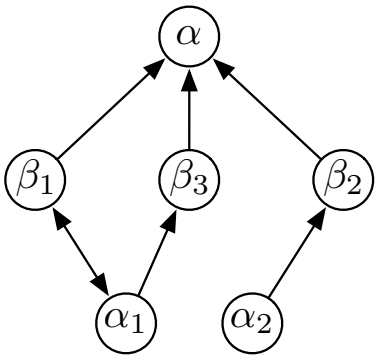

*If there is no other argument for our claim, we add* $\alpha$ *to the set of relevant arguments and continue. Then we move to the next iteration and search for defeaters of* $\alpha$*. To do this, we search for every argument with claim* $\sim At(P,L_2,do(push(R,P,L_2),S_0))$*,* $\sim Free(L_2,S_0)$ *or* $\sim Reachable(P,S_0)$*. Let the following argument being generated in this iteration:*

- $\beta_1 = \langle \{\sim Reachable(P,S_0)\}, \sim Reachable(P,S_0)\rangle$,
- $\beta_2 = \langle \{\sim At(P,L_2,do(push(R,P,L_2),S_0)) \prec \sim At(P,L_2,S_0), not\, Free(L_2,S_0),$
  $\sim At(P,L_2,S_0)\}, \sim At(P,L_2,do(push(R,P,L_2),S_0))\rangle$, *and*
- $\beta_3 = \langle \{\sim At(P,L_2,do(push(R,P,L_2),S_0)) \prec \sim At(P,L_2,S_0), not\, Reachable(P,S_0),$
  $\sim \{At(P,L_2,S_0)\} \sim At(P,L_2,do(push(R,P,L_2),S_0))\rangle$

*Next, we add the arguments* $\beta_1$, $\beta_2$ *and* $\beta_3$ *to the set of relevant arguments and continue by searching for their defeaters. We can generate the following arguments attacking the assumptions of* $\beta_2$ *and* $\beta_3$ *and the premise of* $\beta_1$*:*

- $\alpha_1 = \langle \{Reachable(P,S_0)\}, Reachable(P,S_0)\rangle$
- $\alpha_2 = \langle \{Free(L_2,S_0)\}, Free(L_2,S_0)\rangle$

*We add these arguments to the set and finish the process as there is no additional relevant arguments, since argument* $\beta_2$ *is already in the set. The argumentation framework containing the arguments that are relevant to the query is depicted by the following argumentation graph:*




Note that the above example assumes that all arguments are equally preferred, unifying the defeat and the attack relations. A different ordering would reduce the defeats in the above figure, since one of $\beta_1$ and $\alpha_1$ would be preferred.

Argument generation is based on the search for a suitable defeasible derivation. This process can be conducted using a backward-chaining mechanism to construct a proof tree. A completed tree corresponds to an argument, provided that it does not contain contradicting literals. The backward-chaining process must unify an open subgoal (i.e. a belief that needs to be derived) to the head of a defeasible rule from the agents' theory, add new subgoals from the body of the rule, and continue until no

**Algorithm 3:** Algorithm for the query $\mathcal{D}|\!\approx L(S)$, where $\mathcal{D}$ is a defeasible action theory, $L$ a ground fluent literal and $S$ a ground situation term

Build $AF_{Args_{L(S)}} = \langle Args_{L(S)}, Defs_{Args_{L(S)}} \rangle$;
Generate $GE_{Args_{L(S)}}$ of $AF_{Args_{L(S)}}$;
**if** *there exists* $\alpha \in GE_{Args_{L(S)}}$ *of* $AF_{Args_{L(S)}}$ *with* $Claim(\alpha) = L(S)$ **then**
    **return** true;
**else**
    **return** false;

open subgoals remain. The queries are ground, and so would be the beliefs that need to be matched to the heads of defeasible rules. Rules which have the same literal as their head can be either grounded and then matched, or we can unify them on demand; we unify the relevant terms, but not any other variables appearing in the body of the rule. The backward-chaining mechanism asserts that we exclusively ground rules that can potentially form the support of arguments. Unfortunately, the set of the generated ground rules is a superset of $ground(\mathcal{D})_{Args_{L(S)}}$. This is the case, because the backward-chaining mechanism may explore paths that do not result in the construction of arguments. Grounded beliefs that are generated while exploring such paths may not be part of $ground(\mathcal{D})_{Args_{L(S)}}$ as they are not used by arguments in $Args_{L(S)}$.

Defeaters for an argument are constructed through argument generation for claims that contradict beliefs that appear in the claim or the support of the argument. Essentially, in order to generate all defeaters for an argument $\alpha$ we generate every argument $\beta$ such that $Claim(\beta) \in \{Claim(\alpha)\} \cup Support(\alpha)$.

We consider domains including finite objects and ground situation terms representing plans of finite sequences of actions. These assumptions imply that the ground theory contains a finite number of rules, and as a result, a finite number of arguments can be generated. Accordingly, the argument generation (and the defeater generation) tasks always terminate, given that a cycle detection mechanism is implemented within the defeasible derivation search mechanism, since the process of labelling finite graphs always terminates.

The soundness of Algorithm 3 follows from our definition of warrant, since Algorithm 3 will return 'true' if and only if there exists an argument that has the literal of the query as its claim can be constructed from the theory, and that this argument is included in the grounded extension. Completeness, on the other hand, can be ensured

if exhaustive search is conducted.

Algorithm 3 generates every subgraph relevant for the queried literal, and then uses a labelling mechanism to evaluate argument acceptability. Such subgraphs are not necessarily trees as they may contain arguments defeating multiple other arguments in the subgraph. However, they can be transformed to trees organised in layers. Layer 0 contains the root argument, which claims the queried literal. Layer 1 contains the children of the root node, which correspond to the arguments defeating the root argument. The children of every layer 1 node represent the arguments defeating the argument that corresponds to this node. Every node on an even-numbered layer supports the root argument, whereas every node on an odd-numbered layer defeats the root argument or one of its supporters. The resulting tree contains all arguments that appear in the subgraph. However, arguments that create multiple defeats correspond to multiple nodes, one for every defeat.

Generating such trees in a depth-first manner is advantageous, since in certain cases it is possible to evaluate the acceptability of the root argument before the construction of the entire argumentation tree. This process is performed systematically for abstract argumentation in two-party disputes as described in Vreeswijk and Prakken (2000) and Dunne and Bench-Capon (2003). Depending on the argumentation semantics, rules apply which ensure the correctness of the conclusions. For instance, for grounded acceptability semantics, in order to assert that an argument cannot be supported by itself, it is prohibited from appearing on to two nodes on even-numbered layers in the same path. We discuss protocols for the evaluation of arguments based on argument games in detail in the following chapter.

#### 4.2.2.3 Situation Stratification Strategy

In the previous section, we discussed arguments that are relevant for the evaluation of a warrant of a ground query. This section focuses on the task of defeater generation, and provides a strategy for guiding the evaluation process, when multiple defeaters exist.

The support of an argument regarding a query of the form $L(S)$, where $S$ involves the application of multiple actions, may contain beliefs related to every situation from the initial situation $S_0$ to the situation $S$. Finding all potential defeaters requires the generation of every argument that claims the complement of a statement that appears in the support of the claim or is the claim of the original argument itself. For an argument $\alpha$, this set is denoted by: $Support(\alpha) \cup \{Claim(a)\}$.

The set $Support(\alpha) \cup \{Claim(\alpha)\}$ is conceptually stratified with respect to the sit-

uation terms appearing in the beliefs in this set. Let $Claim(\alpha)$ be equal to $L(S_n)$, $L$ a ground predicate literal and $S_n$ the ground situation term for the ground sequence of actions $\pi = \langle A_1, A_2, \ldots, A_n \rangle$. The beliefs in the set $Support(\alpha) \cup \{Claim(\alpha)\}$ can be stratified into $n+1$ sets $\mathcal{S}_i$, with $i = [0, n]$ containing the beliefs in $Support(\alpha) \cup \{Claim(\alpha)\}$, with a situation term $S_i$. Non-fluent beliefs are placed in $S_0$ together with initial situation beliefs.

According to Proposition 5 on page 76, and Proposition 6 on page 76, the derivation and warrant state of a ground literal referring to a ground situation term $S$ depends entirely on ground beliefs with situation terms that are predecessor or equal to $S$. The specification of successor state axioms (and state constraints/ramifications) asserts that the defeasible derivation of a belief regarding a ground situation $S$ never involves beliefs with successor situation terms. Therefore, while evaluating the acceptability of an argument whose claim refers to the situation term $S$, we do not need to consider arguments whose claims refer to successor situations.

The notion of a sub-argument is very important in this process. Arguments may have multiple defeaters, some of which may be sub-arguments of others. We can utilise this relation while evaluating the acceptability of an argument. There is a clear benefit in initially considering defeats on earlier situations, starting from non-fluent literals and initial situation beliefs. The reason for this is that the acceptability of literals regarding future situations depend on literals referring to their predecessor situations. As a result, knowledge regarding their status can help prune the search.

For example, consider an argument $\alpha$ which has a sub-argument $\alpha'$, and let $\beta$ be an argument defeating $\alpha'$. If $\alpha'$ cannot be defended against $\beta$'s attack, then we know $\alpha$ cannot be defended either.

**Proposition 15.** *Consider an argumentation framework AF and two arguments* $\alpha$ *and* $\alpha'$ *from this framework, such that* $\alpha'$ *is a sub-argument of* $\alpha$*. Let* $GE_{AF}$ *be the grounded extension of AF. If* $\alpha' \notin GE_{AF}$ *then* $\alpha \notin GE_{AF}$*.*

*Proof.* Since $a' \notin GE_{AF}$, there exists at least one argument $b$ such that $b$ defeats $a$ and there does not exist $c \in GE_{AF}$ such that $c$ defeats $b$. If the contrary was the case then it holds that $b \in GE_{AF}$. Also, from the definition of the defeat relation since $a$ is a sub-argument of $a'$ and $b$ defeats $a'$, it holds that $b$ defeats $a$ as well. Therefore, $a$ is not part of $GE_{AF}$. □

We utilise these observations in the next algorithm for argument acceptability checking. Algorithm 4 utilises the situation stratification strategy to potentially reach con-

**Algorithm 4:** The situation stratification strategy for answering queries of the form $\mathcal{D} \mid\!\approx L(S)$. $\mathcal{D}$ is a defeasible action theory, $L$ a ground fluent literal and $S$ a ground situation term

**repeat**
  $AF := \langle \emptyset, \emptyset \rangle$;
  Generate new argument $\alpha$ such that $Claim(\alpha) = L(S)$;
  Add $\alpha$ to $AF$;
  $\mathcal{S} := Support(\alpha) \cup \{Claim(\alpha)\}$;
  Stratify $\mathcal{S}$ in $n+1$ layers;
  **foreach** *layer* $\mathcal{S}_i$ *with* $i = [0, n+1]$ **do**
    Extend $AF$ with $Args_{L(S)}$ from theory $ground(\mathcal{D}, S_i)$ ;
    Extend $GE$ for current $AF$;
    **if** $\alpha \notin GE$ **then**
      break;
  **if** $\alpha \in GE$ **then**
    **return** true;
**until** *there exists no new argument* $\alpha$;
**return** false;

clusions about warrants without evaluating every defeater. The basic idea behind the algorithm is that the points at which a literal can be attacked are literals which can be stratified according to their situation terms.

We build the argumentation graph in phases, each of which introduces arguments which claim literals with a specific situation term. The initial phase considers arguments about non-fluent and fluent literals about the initial situation. Every following phase extends the graph with arguments regarding the successor situation. The final phase introduces arguments about the same situation as the literal of the query. In every phase, after the expansion of the graph we extend the labelling to the newly added arguments. The labelling information of the already existing arguments, apart from $\alpha$, is not affected by the added arguments. All new defeats are the product of sub-arguments of the new arguments, which have already been considered in the previous labelling process.

Until we reach the final phase, the labelling process cannot determine if $\alpha$ is defeated against every potential defeater. However, if at the end of any phase $\alpha$ is cannot

**Algorithm 5:** An exhaustive breadth-first planer

```
l := 0;
while l < ε do
    repeat
        Select a new plan π of length l;
        if D |≈ goal(S_π), executable(S_π) then
            return π;
    until there does not exist a new plan of length l;
    l++;
return null;
```

be defended against a defeater, then we can derive that $\alpha$ is not part of the grounded extension. the arguments that would be added by the following phases, would not affect the labelling of the defeater. Therefore, if such a defeater exists, we can determine that $\alpha$ is not acceptable without proceeding to the next phase.

The argument generation step can reuse derivations that have been made in the previous phases. The exhaustive nature of argument and defeater generation implies that Algorithm 4 is sound and complete.

#### 4.2.2.4 Executability-Based Plan Space Pruning

Algorithm 1 does not provide any insight on the order in which plans are evaluated. Based on plan length, we distinguish the following basic strategies: *breadth-first* and *depth-first* planning.

An exhaustive breadth-first search strategy evaluates all plans of an increasing size, starting from *zero* and gradually reaching a threshold $\varepsilon$. More specifically, it evaluates whether the expression $goal(S_\pi), executable(S_\pi)$ is warranted for situation term $S_0$. Afterwards, it progresses with the evaluation of the statement for every situation $S_1 = do(A, S_0)$ such that $A$ is a ground action. If no warranted plan is found, it continues with their successor situations. The search continues in the same manner, until situations $S_\varepsilon = do([\pi_\varepsilon], S_0)$, where $\pi_\varepsilon$ is a plan of $\varepsilon$ actions. The main advantage of breadth-first search is the guarantee that the shortest warranted plan within the selected threshold is always identified. Algorithm 5 describes a breadth-first planner.

An exhaustive depth-first strategy begins with the evaluation of the empty plan. If this is not warranted, it proceeds by selecting an action and evaluating the respective

**Algorithm 6:** An exhaustive depth-first planer

**repeat**
  $\pi := \emptyset$;
  **repeat**
    Construct plan expression $goal(S_\pi), executable(S_\pi)$;
    **if** $\mathcal{D} |\!\approx goal(S_\pi), executable(S_\pi)$ **then**
      **return** $\pi$;
    Select a ground action $A$ such that the sequence $\pi, A$ has not been evaluated;
    $\pi := \pi, A$;
  **until** *length*$(\pi) > \varepsilon$ *or no new plans exist*;
**until** *no new plans exist*;
**return** null;

plan. If this also fails to be warranted, another action is appended, and the new plan is then evaluated. This process is repeated until a warranted plan is discovered or the length of the plan exceeds a certain threshold. In this case, we backtrack and search again making different action choices. A depth-first planning algorithm is outlined by Algorithm 6.

The depth-first search does not necessarily return the shortest plan. However, this strategy is preferable for large domains because it enables the reuse of conclusions and provides a simple pruning mechanism based on situation executability. This mechanism follows the observation that the derivation and warrant results regarding an action sequence $\pi$ are also relevant for every sequence $\pi'$ extending $\pi$. Therefore, negative results regarding the executability of a situation carry over to every successor situation.

**Observation 5.** *Let $\mathcal{D}$ be a defeasible basic action theory and $S$ be a ground situation term. For every situation $S' = do([A_1, \ldots, A_n], S)$ extending $S$, if $\mathcal{D} |\!\not\approx executable(S)$ then $\mathcal{D} |\!\not\approx executable(S')$.*

This observation follows from the executability abbreviation and the definition of the warrant relation. The abbreviation $executable(S')$ is expanded to:

$$executable(S), Poss(A_1, S), \ldots, Poss(A_n, do([A_1, \ldots, A_n], S)) .$$

From the definition of warrant, it follows that a conjunction is warranted if every literal in the conjunction is warranted. Therefore, since $\mathcal{D} |\!\not\approx executable(S)$ it holds that $\mathcal{D} |\!\not\approx$

$executable(S')$.

Argumentation in our framework is performed as two-step process: searching for defeasible derivations, and evaluating the acceptability of the corresponding arguments. As a result, the warrant relation is based on the notion of defeasible derivation.

**Observation 6.** *Let $\mathcal{D}$ be a defeasible basic action theory and S be a ground situation term. For every situation $S' = do(\pi, S)$, where $\pi$ is a possibly empty sequence of ground actions, if $\mathcal{D} \not|\sim Poss(S)$ then $\mathcal{D} \not|\sim executable(S')$.*

The observation follows from the executability abbreviation and the definition of the warrant relation. The abbreviation $executable(S')$ expands to:

$$executable(S), Poss(A_1, S), \ldots, Poss(A_n, do([A_1, \ldots, A_n], S)).$$

Regardless of the number of actions in $\pi$, in order for this statement to be warranted, $executable(S)$ must be warranted from the theory. This is not the case, since $Poss(S)$ is not warranted from $\mathcal{D}$, since there exists no argument claiming $Poss(S)$.

Algorithm 7 utilises the above propositions to prune the search for a warranted plan. Whenever a situation $S$ that is not executable is identified, the search restarts for a different situation sequence, since no successor of $S$ is executable either. This algorithm reduces the overall 'amount' of argumentation required, while increasing the need to search for defeasible derivations. Therefore, computationally expensive processes such as generating arguments and finding defeaters are performed exclusively for potential plans.

For readability purposes, we extend the defeasible derivation relation to conjunctive statements. More specifically, $\mathcal{D} |\sim L_1, L_2, \ldots, L_m$ if and only if $\mathcal{D} |\sim L_1$, $\mathcal{D} |\sim L_2$, ..., and $\mathcal{D} |\sim L_m$.

#### 4.2.2.5 Action Selection Heuristic

The depth-first planner presented in the previous section does not provide any insight on the problem of *action selection*. More specifically, if after a plan $\langle A_1, A_2, \ldots, A_m \rangle$ is applicable but does not achieve the goal, the planner picks an action $A_{m+1}$, and proceeds with the evaluation of the plan $\langle A_1, A_2, \ldots, A_{m+1} \rangle$. When multiple options for action $A_{m+1}$ are available, this process is performed non-deterministically.

In large planning domains, the number of ground actions is considerable. Non-deterministic action selection is impractical, since it may lead to the selection of unhelpful or potentially even destructive actions. Exhaustive search eventually leads to a solution, but is infeasible in many cases.

**Algorithm 7:** Depth-first planner with executability-based pruning

**while** *new plans exist* **do**
  **while** $length(\pi) < \varepsilon$ **do**
    $\pi :=$ a new plan $\langle \pi, A \rangle$;
    $S_\pi :=$ situation term for plan $\pi$;
    **if** $\mathcal{D} \not\sim Poss(S_\pi)$ **then** break;
    **if** $\mathcal{D} \sim goal(S_\pi)$ **then**
      **if** $\mathcal{D} \approx executable(S_\pi)$ **then**
        **if** $\mathcal{D} \approx goal(S_\pi)$ **then return** $\pi$;
      **else** break;
  $\Pi_{n-ex} := \Pi_{n-ex} \cup \{$non-executable subsequence of $\pi\}$;
  $\pi :=$ subsequence of $\pi$ such that $\pi \notin \Pi_{n-ex}$;

In order to improve the efficiency of the planning method, we propose a heuristic action selection strategy inspired by the "no delete lists" heuristic from the planning literature (McDermott, 1996; Bonet and Geffner, 2001; Hoffmann and Nebel, 2001). The "no delete lists" heuristic is one of the most successful planning heuristics. The heuristic quality of a state is measured based on the size of a plan that solves a relaxed planning problem, in which delete lists (i.e. the negative effects of actions) are ignored. Solutions to the relaxed planning problem are simpler to calculate, and these solutions have been empirically shown to provide good estimates in benchmark planning domains Hoffmann (2005).

Our axiomatisation of successor state axioms is based on the use of default negation preceding disruptive effects. These axioms have the following form:

$$L(do(a,s)) \multimap \gamma_L(s); L(s), not(\gamma_{\overline{L}}(s)),$$

where $\gamma_L(s)$ abbreviates the body of the compound effect axiom for literal $L$.

Following the structure of these axioms, and the specification of the notion of defeasible derivation, there exists a defeasible derivation $\mathcal{D} \sim L(S')$, where $L(S')$ is a ground literal, $S' = do(A,S)$ is a ground situation term, and $A$ is a ground action, if and only if one of the following conditions holds:

- There exists an effect rule for action $A$ producing $L$ and every literal in the body of the rule can be derived from $\mathcal{D}$.
- $\mathcal{D} \sim L(S)$.

Following the treatment of default negated literals as assumptions, the use of default negation prior to the disruptive effects for $L$, results in defeasibly deriving $L(S')$ when $\mathcal{D}|\sim L(S)$, regardless of the derivation status of the literals of the disruptive effects.

As a result, defeasible derivations can be made for the literals produced by the final action, or any one of the previous actions leading to the current situation. Disruptive effects of actions are disregarded. Accordingly, the set of literals that can be derived in a successor situation subsumes the literals derivable from its predecessors.

We introduce the term *goal derivable* situation to represent any situation term $S_{gd}$ such that $\mathcal{D}|\sim goal(S_{gd}), executable(S_{gd})$. The *heuristic value* is a function $h(S)$ that measures the quality of a situation based on the number of actions needed to reach a goal derivable situation from the current situation. If this is not possible, then the function takes an arbitrary high integer value.

$$h(S) = \begin{cases} \min_{S^* \in \mathcal{S}_{gd}}(|S^*|) - |S| & \text{if } \mathcal{S}_{gd} \neq \emptyset \\ \max \text{int} & otherwise \end{cases}$$

$\mathcal{S}_{gd}$ is set of all goal derivable situations that are successor (or equal) to $S$. $|S|$ denotes the distance, in terms of number of actions, between $S$ and the initial situation $S_0$. More specifically, $|S_0| = 0$, $|do(A, S_0)| = 1$ and $|do([A_1, A_2, \ldots, A_m], S_0)| = m$. An outline for the calculation of the heuristic value is provided by Algorithm 8.

**Algorithm 8:** Heuristic Value Calculation

```
l := 0;
S := {S_0};
while l < ε do
    foreach S ∈ S do
        if D|~goal(S) then
            return l;
    l++;
    S := {do(A,S) | S ∈ S, A is a ground action such that D|~Poss(A,S)};
return ε;
```

Our use of the heuristic value follows the strategy behind the popular FastForward planner (Hoffmann and Nebel, 2001). Initially, we search the plan space in a depth-first manner using the *enforced hill climbing* strategy, which greedily moves to the nearest, strictly better situation discovered using breadth-first search. Algorithms 9

**Algorithm 9:** Enforced Hill Climbing Planner

$S^* := S_0$;
$S := null$;
**while** *a threshold* $\varepsilon$ *is not reached and* $S \neq S^*$ **do**
    $S := S^*$;
    **if** $\mathcal{D} |\!\approx G(S)$ **then**
        **return** the plan that corresponds to $S$;
    **foreach** $S'$ *successor to* $S$ *such that* $\mathcal{D} |\!\approx Executable(S')$ *and* $h(S') < h(S^*)$ **do**
        $S^* := S'$;
        break;
**return** *null*;

and 10 outline the enforced hill climbing and best-first search strategies respectively.

Enforced hill climbing search is not complete, since it greedily moves towards situations of higher heuristic quality. When it reaches a local maximum, that is a situation with a better heuristic quality than any successor situations, that does not satisfy the goal, it fails. In situations in which enforced hill climbing fails, the heuristic value is still helpful, as it can guide the search by prioritising actions leading to situations of higher heuristic quality (although in this case we calculate it using $\min_{S^* \in \mathcal{S}_{gd}}(|S^*|)$ rather than $\min_{S^* \in \mathcal{S}_{gd}}(|S^*|) - |S|$ to avoid unsuccessful local maxima). In this case search is performed in a *best-first* manner. Best-first search explores the plan space by expanding the most promising, in terms of the specified heuristic value, reached situation term. Its exhaustive nature guarantees that, if a solution exists within a certain length, it will be eventually discovered.

### 4.2.3 Summary

The extensive size of ground DBATs makes simple, propositional argumentation-based approaches highly impractical. In order to tackle this problem, we do not treat situation variables in the same manner as object variables with respect to grounding. We transform the planning algorithm so that it revolves around queries of the form $\mathcal{D} |\!\approx L(S)$. Such queries can be answered by grounding the theory for situation terms $\mathcal{S} = \{S' \mid S' \text{ is ground situation term that is predecessor or equal to } S\}$.

Depth-first search planning is suitable for searching the plan space, since it is based

**Algorithm 10:** Best-First Search Planner

**if** $\mathcal{D} \mid\!\approx goal(S_0)$ **then**
  **return** *empty plan*;
$\Sigma := \{S_0\}$;
$i := 0$;
**while** $i < \varepsilon$ **do**
  $S := \arg\min_{S' \in \Sigma}(h(S'))$;
  $\Sigma := \Sigma \setminus \{S\}$;
  **foreach** *ground action A such that* $\mathcal{D} \mid\!\approx Poss(A,S)$ **do**
    $S' := do(A,S)$;
    **if** $\mathcal{D} \mid\!\approx goal(S')$ **then**
      **return** plan corresponding to $S'$;
    $\Sigma := \Sigma \cup \{S'\}$;
    $i++$;
**return** *null*;

on the expansion of a plan $\pi$ with an additional action $A$ in every step. Accordingly, conclusions made regarding derivations and the warrant state of literals after the application of $\pi$ are relevant for $\pi' = \langle \pi, A \rangle$. If we identify sequences that are inapplicable, we prune the search space as any plan extending them is also inapplicable. In order to perform an equivalent mechanism during breadth-first search, we would have to store all argumentation results that are relevant to every situation that can be expanded in the following iteration. This task is significantly complex in terms of memory requirements.

However, the depth-first search mechanism does not provide insight on action selection, and in any realistic domain there are multiple options for every action selection step. To this end we followed a heuristic approach. The heuristic value acts as a guide in the search for potential plans, reducing the possibility of selecting unnecessary or even potentially harmful, actions. The heuristic value needs to be calculated quickly, since it is measured for every potential transition. Based on the defeasible implication relation and the use of default negation, we calculate the heuristic value based on a relaxed version of the planning problem. The solution to the relaxed problem is based on defeasible derivations, rather than the expensive argumentation task, which is only performed for actions with high heuristic quality. Hill climbing search may not lead to

any solutions, since it only takes a single path into account. The heuristic value is still helpful as a guide while searching the space using exhaustive best-first search.

In contrast to classical planning approaches that search the state-space of the planning domain, our methods search the situation space. The benefit of searching the state space is that, since the same state may be reachable by different plans, it is possible to prune the search whenever it proceeds in circles. Unfortunately, our problem is bound to the notion of situations and has different semantics. Even if exactly the same literals can be derived in two different situations, their warrant status may be different. The next section focuses on the differences between MPCP with our set-theoretic formalism and classical planning, and proposes methods for synthesising warranted plans that delegate the search for potential plans to efficient, state-of-the-art planners, in order to optimise the efficiency of the overall process.

## 4.3 Planning with MPCP Problems

Multi-perspective cooperative planning consists of the planning problem of synthesising potential plans, and the decision-making problem of evaluating these plans against possible objections. This section focuses on algorithms for the solution of the problem based on the proposed set-theoretic notation.

The planning problem is essentially the problem of synthesising candidate plans. In order to provide an overall solution to the problem based on the set-theoretic notation, we need to provide a suitable specification of the decision sub-problem based on the MPCP formalism. To this end, we introduce the notion of acceptability based on MPCP problems and a preference ordering over planning beliefs, emulating the notion of warrant on plans.

In addition, we focus on the planning sub-problem of MPCP. More particularly, we identify its main differences from classical planning. We provide methods for the transformation of the problem into a classical planning theory, suitable for delegating the synthesis of candidate plans to standard planning algorithms. Finally, we provide algorithms which utilise efficient, state-of-the-art planners, for the construction of candidate plans, exploiting their heuristic and highly optimised mechanisms for searching the state space of a planning domain.

### 4.3.1 MPCP-Based Argumentation

In order to represent and provide a solution to the decision-making part of the problem using the MPCP formalism, our approach must account for the following:

- Specification of the structure of arguments.
- Specification of the defeat relation among arguments.
- Performing the task of argument acceptability evaluation.
- Performing the task of plan warrant evaluation.

Contrary to the aforementioned defeasible situation calculus argumentation-based methods, the set-theoretic notation does not offer a logic-based inference mechanism. As a result, argument structures are specified in an ad-hoc manner which is based on the state transition function, which is the main reasoning mechanism provided in MPCP.

Contrary to our previous work with DBATs, argumentation methods based on MPCP are based on the notion of a state. When we traverse through a plan, we identify the state transitions caused by its actions, and gather reasons explaining the values of literals in the resulting states. This task is called *plan projection*, and is the basis of argument generation in MPCP-based argumentation.

#### 4.3.1.1 Plan Projection

Plan projection projects the effects of an action sequence on the state of the environment. It iterates over the actions in the plan, starting from the initial state. First, it calculates whether this state satisfies the preconditions of the action. This is performed by searching whether there exists an operator specification, according to which every precondition is satisfied by this state. Then it calculates the successor state based on the state transition function. This process continues for every action in the sequence. When every action has been applied, the process checks whether the agents' goals are satisfied in the resulting state. Plan projection fails if there exists an inapplicable action in the plan or if the goal literals are not satisfied by the final state.

The results of plan projection can be represented as a directed graph, whose nodes are organised in layers of two different types. Odd-numbered layers correspond to states. The nodes in these layers represent literals, and are denoted by circles. Layer 0 and every even-numbered layer is a justification layer. Every justification corresponds to a single-step derivation describing the reasons to believe that a literal holds in a state,

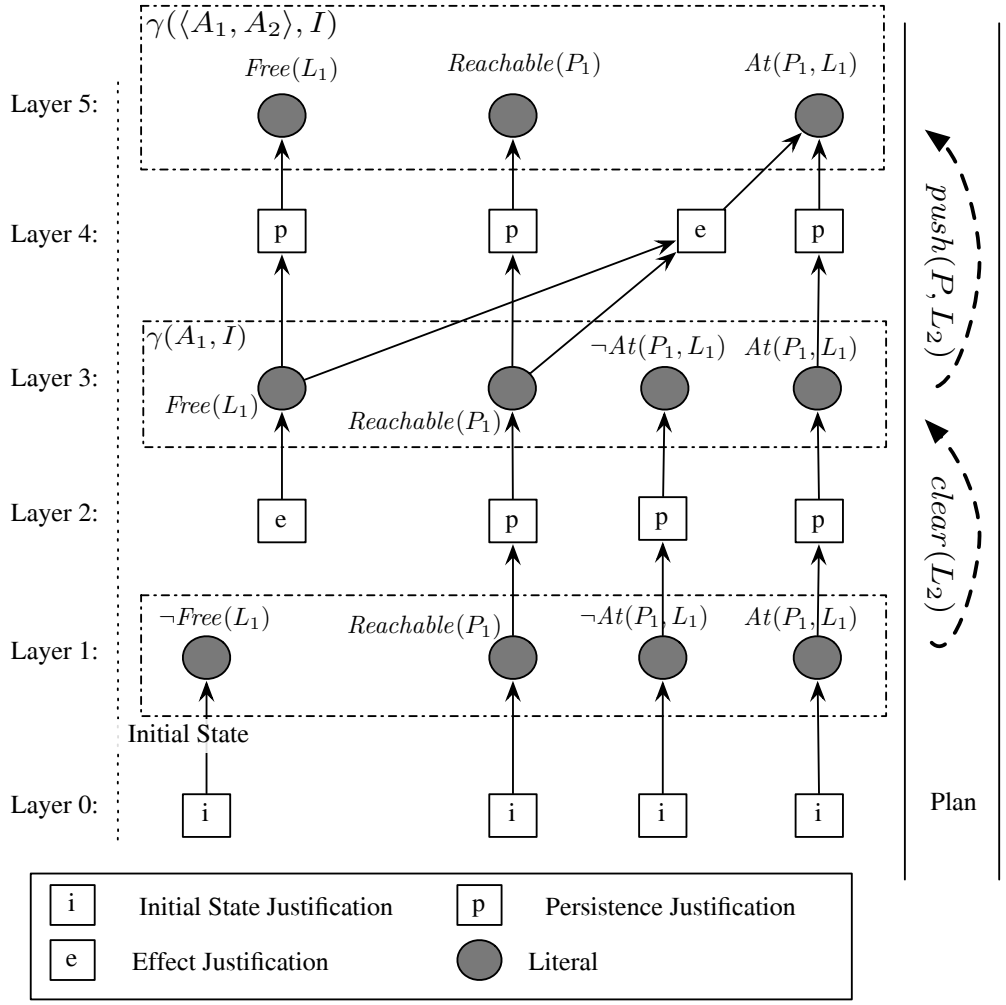



Figure 4.1: Directed graph representation of plan projection data

based on the relevant conditions in its predecessor state. Justifications are denoted in the graphs using squares. An example of a plan projection graph is depicted in Figure 4.1.

We differentiate between the following classes of justification:

- Initial state justification
- Persistence justification
- Effect justification

Initial state justifications are equivalent to defeasible initial state axioms. For every literal that is part of the initial state we generate an initial state justification. Initial state justifications occupy layer 0, and do not have any incoming edges.

Persistence justifications correspond to derivations made using the frame part of defeasible successor state axioms. They describe that the agents have reasons to believe that the latest action did not affect the current state of a literal. Formally, there exists a persistence justification for every literal $L$ in state $\gamma(A,\sigma)$ if and only if there exists an agent $i$ such that for every conditional effect $\langle \mathcal{C}_{\overline{L}}, \overline{L} \rangle \in \mathit{eff}_i(A)$, it holds that there exists $C_{\overline{L}} \in \mathit{pre}_i(A) \cup \mathcal{C}_{\overline{L}}$ such that $C_{\overline{L}} \notin \sigma$ or $\overline{C}_{\overline{L}} \in \sigma$.

Effect justifications denote that a literal is added to a state because there exists an applicable effect producing this literal. These correspond to derivations made from the effect part of the ground defeasible effect axioms. Formally, there exists an effect justification for literal $L$ in state $\gamma(A,\sigma)$ if and only if there exists an agent $i$ with a conditional effect $\langle \mathcal{C}_L, L \rangle \in \mathit{eff}_i(A)$, such that $\mathit{pre}_i(A) \cup \mathcal{C}_L \subseteq \sigma$. An edge starting from an effect justification connects it with the literal it justifies on the following layer. Incoming edges to the justification connect to it every condition in $\mathit{pre}_i(A) \cup \mathcal{C}_L$ in the predecessor state. If the effect is not conditional and the action producing this effect has no preconditions, then the effect justification node has no incoming edges.

#### 4.3.1.2 Arguments

Plan projection graphs contain sufficient information to identify derivations and construct arguments. Every literal node in the graph corresponds to a literal, related to the corresponding action sequence. This is similar to literals grounded with respect to a situation term corresponding to a plan in defeasible reasoning with DBATs. There may exist multiple derivations for a literal. The *compound defeasible derivation graph* conveys this information.

**Definition 29.** *Given a plan projection graph $\mathcal{G}$ and a node literal L, a* compound derivation graph *$\mathcal{G}_L$ is the subgraph of G which contains every node $L'$ for which there is a path from $L'$ to L and every arc from $\mathcal{G}$ among these nodes.*

Figure 4.2 depicts a compound derivation graph for a literal $L$. This graph represents every possible way that the literal can be derived. A *derivation* corresponds to a subgraph of this graph which contains a single justification node connected to a literal node on the exactly higher layer. Compound derivation graphs may have multiple justifications for the same literal.

**Definition 30.** *Given a compound derivation graph $\mathcal{G}_L$ for a literal L, a* derivation graph *for L is a subgraph of $\mathcal{G}_L$ which contains a justification node connected to every*

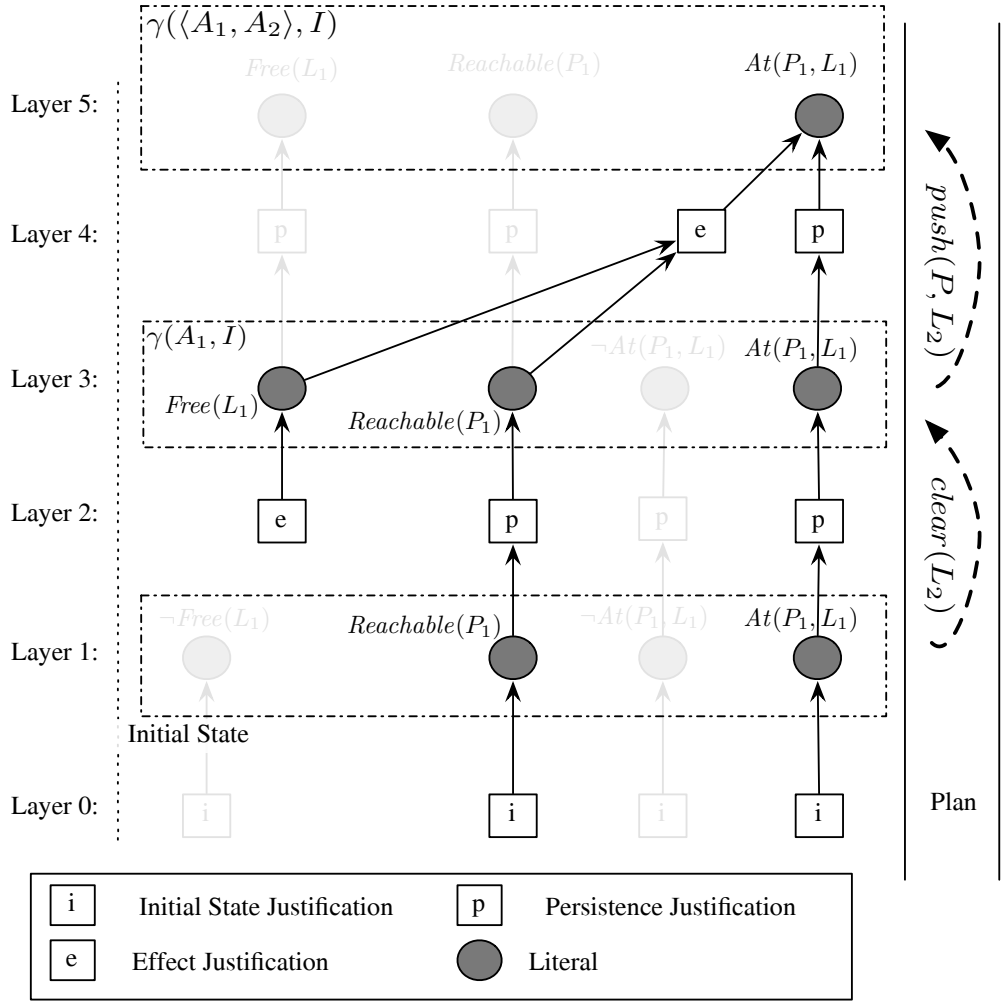



Figure 4.2: Compound derivation graph for literal $L$ on the state represented by layer 5 constructed using plan projection data

*literal node. Nodes and edges that are not part of a path leading to L are not part of the graph.*

*Arguments* correspond to derivations, but also require the literal layers to be non-contradictory. This poses an additional task that needs to be executed over the task projection data to disqualify derivations made using contradicting literals in the same state. Similar to the formal argument definitions for arguments in defeasible logic, these graphs are minimal, in the sense that if we omit any edges or nodes it is impossible to reach the conclusion, and their premises are based on beliefs from the agents' theory, in terms of initial state beliefs and the operator specification that formulated the justifications.

The compound derivation graph corresponds to compound argument graphs, which include every possible argument that can be put forward for the respective claim. From these graphs we must exclude every derivation which is based on contradictory literals within the same layer. Compound arguments may contain multiple justifications of the same literal in the same layer.

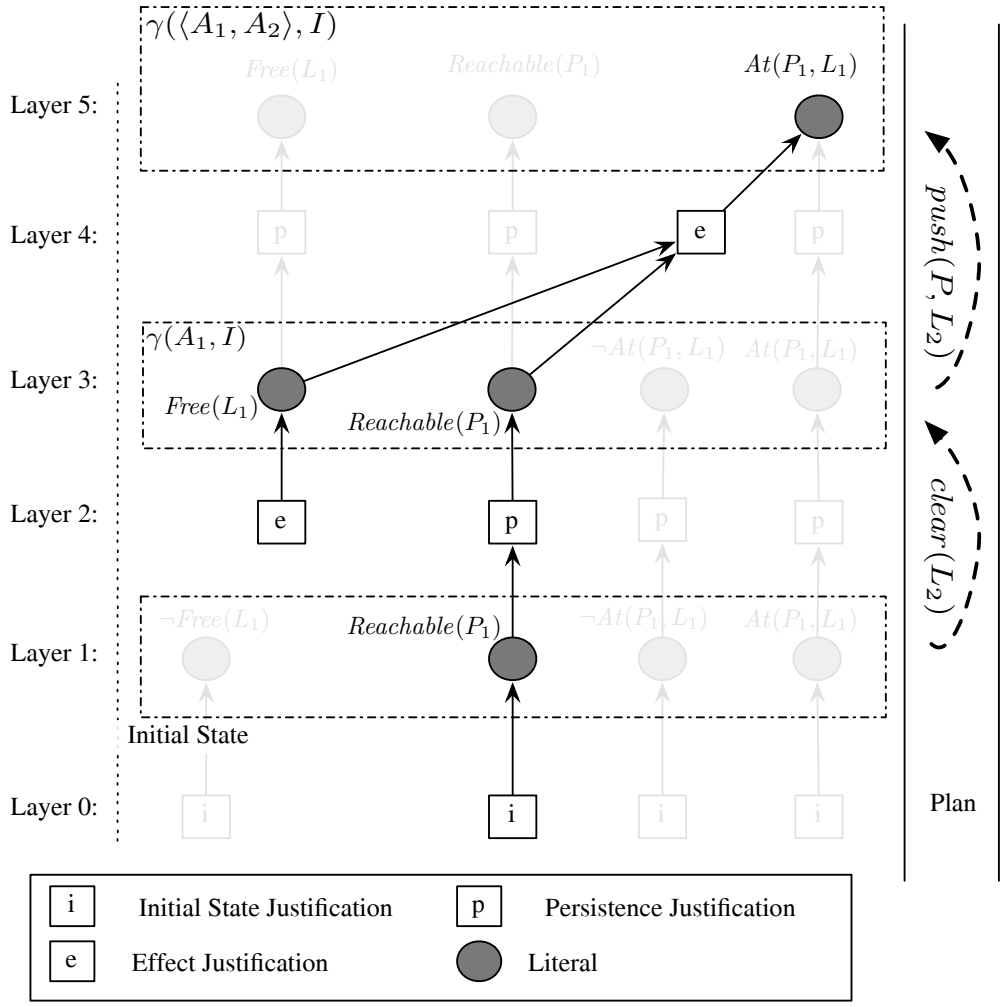



Figure 4.3: An argument claiming that $L$ holds on the state represented by layer 5

Arguments are subgraphs of the compound derivation graphs. To construct these, we start from the derived literal in the top literal layer $k$ and add it to the argument subgraph. Then, we move to the justification layer $k$ and add one justification node $j$ linked to the literal added in $k$ to the argument graph. Afterwards, we proceed to layer $k-2$ and add every literal connected with justification $j$ in layer $k-1$ to the argument graph. This process continues, by selecting one justification for every literal in the

higher layer from the layer below, and every literal for each justification in the higher layer, until there are no other nodes to be added (i.e. justifications with no conditions have been reached).

*Attacks*, similar to defeasible logic programming, are based on the notion of sub-argument. An attack is defined as a disagreement between a literal regarding a layer and its negation in the same layer. This can be the claim of the argument or the claim of a sub-argument of the main argument in the graph. A sub-argument is the sub-graph of the graph representing an argument, containing the claim of the sub-argument, and every node for which there is a path connecting it to this literal.

*Defeats* are calculated based on the attack relation and specific domain dependent principles, similar to the defeasible basic action theories case. Given an equivalent preference ordering over arguments that is coupled with two corresponding DBAT and MPCP problems, both theories provide equivalent acceptability results.

For example, a preference ordering over initial state beliefs and operator specifications may be used to calculate a preference ordering over arguments based on the belief with minimal preference used for the derivation of the argument's conclusion. Initial state justifications receive the preference value of initial state beliefs. Persistence justifications receive the preference of the literal in the previous state. Finally, effect justifications receive preference of their least preferred condition or the relevant action specification if it has a lower preference than the least preferred condition.

Arguments can be represented as tuples of the form $\langle \mathcal{B}, h \rangle$. $\mathcal{B}$ is the support of the argument and $h$ represents the argument's claim. Each element of the set $\mathcal{B} \cup \{h\}$ is a triple of the form $\langle L, k, p \rangle$, where $L$ is literal, $k$ is the layer in which the literal appears and $p$ is the preference value for the derivation of the literal in this layer. Arguments are non-contradictory, and as a result the support sets cannot contain contradicting elements such as $\langle L, k, p \rangle, \langle \overline{L}, k, p' \rangle$.

In this setting, we define a relation between the arguments constructed from a MPCP problem (which follows the initial state completeness assumption) and the arguments generated from the corresponding DBAT.

**Definition 31.** *Consider two arguments* $\alpha$ *and* $\beta$*,* $\alpha$ *constructed from a DBAT and* $\beta$ *generated from the corresponding MPCP problem. We say that* $\alpha$ *corresponds to* $\beta$ *(and* $\beta$ *to* $\alpha$ *equivalently) if and only if:*

- $Claim(\alpha) = Claim(\beta)[S]$*, where S is the ground situation term corresponding to the actions leading to the layer of* $Claim(\beta)$.

- *Every sub-argument of* $\alpha$ *corresponds to a sub-argument of* $\beta$.
- *Every sub-argument of* $\beta$ *corresponds to a sub-argument of* $\alpha$.

The arguments constructed from a *DBAT* subsume the arguments constructed from the corresponding MPCP problem.

**Proposition 16.** *Let a MPCP problem* $P = \langle N, F, I, O, G \rangle$*, and the corresponding DBAT* $\mathcal{D} = \langle \mathcal{D}_{ss}, \mathcal{D}_{ap}, \mathcal{D}_{S_0}, \mathcal{D}_{una}, \mathcal{D}_c \rangle$*, both with complete initial states. Also, let* $\alpha$ *be an argument that can be constructed from P. There exists a corresponding argument* $\beta$ *that can be constructed from* $\mathcal{D}$ *with claim* $h[S]$*, and every sub-argument of the later corresponds to a sub-argument of* $\langle \mathcal{B}, h \rangle$.

*Proof.* Proof by induction on the layer $k$ of the claim $Claim(\alpha) = L$.
(Base Case) $k = 1$, then the argument claims an initial state belief. From the translation mechanism of MPCP problems to DBATs, we know that if $L \in I$ then $L(S_0) \prec \in \mathcal{D}$. Therefore, there exists an argument claiming $L(S_0)$. None of the arguments have any sub-arguments.
(Induction Step) We assume that the proposition holds for $k = n$. We show that it holds for the following literal layer $n+2$.

> Let $k = n+2$. Since there exists an argument for $L$ in layer $n+1$, then $L \in \gamma(\langle A_1, \ldots, A_m \rangle, \sigma)$, where the sequence of action $A_1, \ldots A_m$ leads to the state represented by layer $n+2$.
>
> From Proposition 10, we know that the defeasible derivations made using a BAT subsume derivations made using the state transition function in MPCP problems. As a result, since $L \in \gamma(\langle A_1, \ldots, A_m \rangle, \sigma$, it holds that $\mathcal{D} \in L(S_m)$, where $S_m$ is the situation term corresponding to $A_1, \ldots A_m$.
>
> The final derivation step is made either using a ground defeasible effect or a defeasible frame axiom. In both cases, every literal in the body of the rule refers to situation $S_{m-1}$, where $S_m = do(A_m, S_{m-1})$. These literals are part of $\gamma(\langle A_1, \ldots, A_{m-1} \rangle, \sigma)$. For every such literal $L'$ it holds that $\mathcal{D} \mid\sim L'(S_{m-1})$.
>
> According to the assumption of the induction step, there exists a corresponding argument that can be constructed from $\mathcal{D}$, with the same claim and sub-arguments, for every argument of layer $n$ (which is the layer where the beliefs regarding situation terms).

As a result, for every such literal $L'$, there exists a sub-argument that can be constructed from $\mathcal{D}$. Accordingly, there exists an argument that can be constructed from $\mathcal{D}$ with claim $L(S_m)$, and every sub-argument of the later corresponds to a sub-argument of $\beta$ with claim $L$ in layer $n+2$.

□

The use of default negation in DBATs leads to the construction of additional arguments that cannot be constructed from the corresponding MPCP problem. These arguments are not acceptable.

**Proposition 17.** *Let a MPCP problem P, and the corresponding DBAT $\mathcal{D}$, both with complete initial states. For every argument $\alpha$ that is constructed from $\mathcal{D}$ and that is acceptable with respect to grounded argumentation semantics, it holds that there exists a corresponding argument $\beta$ that can be constructed from P.*

*Proof.* Proof by induction on $k$ length of the situation term $S$ for $Claim(\alpha) = L(S)$.
(Base Case) $k = 0$. Since $\mathcal{D} \mid\approx L(S_0)$, we infer that $\mathcal{D} \mid\sim L(S_0)$. From the theory translation mechanism from MPCP problems to DBATs, we derive that $L \in I$. As a result, there exists a corresponding argument form $P$. Neither of the arguments has any sub-arguments.
(Induction Step) We assume that the proposition holds for $k = n$, with $S_n$. We show that it holds for $k = n+1$, in situation $S_{n+1}$.

From $\mathcal{D} \mid\approx L(S_{n+1})$ and Proposition 12, we derive that $L \in \gamma(\pi_{n+1}, I)$, where $\pi_{n+1}$ is the plan corresponding to situation term $S_{n+1}$. The derivation that is used for the acceptable argument claiming $L(S_{n+1})$ may be based either on a ground defeasible frame or an effect axiom for the final derivation step. In both cases, every literal, apart from the ones preceded by default negation, is warranted. As a result, following the assumption of the induction step, there exists an argument from $P$ for each one of these. The MPCP argument generation method ensures that there exists an argument for every literal that is part of a state, if the relevant conditions (in the case of an effect rule) or the same literal (in the case of inertia) are part of the previous state. Therefore, there exists an argument for $L$ in the layer corresponding to situation $n+1$. □

Equivalence of the warrant results from DBATS and MPCP problems depends on the mechanism that links conditions with persistence justification nodes. We have designed two methods for identifying the incoming connections towards a persistence

justification node. The first is focused on practicality, whereas the other emulates inference made using frame axioms in a more accurate manner, but is harder to calculate.

The first approach adds an edge from the literal being justified, in the previous layer, to the persistence justification node. This is similar to an argument generated from a DBAT that is based on a single step derivation that follows a ground frame axiom, but disables attacks on default negated literals. Note that we only introduce persistence justifications if and only if there is an action specification that entails them, so this method does not introduce arbitrary arguments. However, this inference scheme is biased towards persistence, compared to the warrant specification in DBATs. As a result, there may be cases in which according to a MCPC a literal $L$ is warranted in a state $\sigma$ (because of persistance), but on the other hand neither $L(S)$ or $\overline{L}(S)$ are warranted by the corresponding DBAT (where $S$ is the situation that corresponds to the plan to reach the state $\sigma$). Apart from this, the completeness properties of the approach is preserved. Consider the following example.

Let $\langle\{\neg Broken(L)\}, Lit(L)\rangle$ be a conditional effect of the action $switch\_on(L)$ and a lamp $L$. Assume that the initial state includes both $Broken(L)$ and $\neg Broken(L)$, and that they are equally preferred. As a result, the justification for the effect condition is not acceptable, since it cannot be defended against an attack on the premises used to derive the conclusion. Therefore, if $\neg Lit(L)$ is acceptable in the state prior to the application of the action, then $lit(L)$ is also acceptable after the application of $switch\_on(L)$. This does not hold for the corresponding DBAT, since there is not adequate defence against the attack on the assumption $not\,\neg Broken(L)$ which appears on the successor state axiom for $\sim Lit(L)$.

We designed an alternative method that overcomes this issue at the expense of increasing the required amount of reasoning effort. This method adds a persistence justification, which justifies literal $L$, for every operator specification and every set of conditions that sufficiently provide reasons to believe that every conditional effect producing $\overline{L}$ within this specification is not applicable. If there exists only one operator specification with a single conditional effect this method is simple. For instance, in the previous example, we only have to add an edge from $Broken(L)$ to the persistence justification which justifies $\neg Lit(L)$. In the general case this process is complex, since we need to consider all combinations of conditions that are specified by a single specification, and may result in the literal remaining unchanged. This process does not increase the number of arguments that are constructed, but increases the premises in the support of arguments that include persistence justifications. The additional premises result in

further attacks between such arguments, since they allow attacks against these conditions. As a result in order for a literal justified by a persistence justification to be accepted, all attacks on conditions of the persistence justification must be defended. This scheme produces the same acceptability results as DBATs in domains following the initial state completeness assumption.

In practice, we can simplify this process slightly by only performing it literals which are added as effects to the successor state, but are not deleted since there is a specification according to which they are unaffected by the final action. Note that preference orderings with respect to persistence justifications must depend exclusively on the preference value of the justified literal in the previous state, not on any added conditions.

#### 4.3.1.3 Labelling Plan Projection Graphs

Argument acceptability is calculated in the same way for argument graphs as for defeasible basic action theory arguments. An argument must be defended against every defeat, according to the relevant argumentation semantics. Algorithm 11 presents a labelling method for labelling the nodes of the plan projection graph in order to identify which literals are warranted. The input of the process is the plan projection graph, and the output is a labelled plan projection graph. Nodes in the labelled graph are labelled $W$ or $N$, according to their warrant information. $W$ denotes that the literal is warranted in the relevant state according to the agents' theory, whereas $NW$ represents the contrary. The labelling process follows the grounded (sceptical) argumentation semantics. If a derivation for a literal is based on contradicting literals in the same state $\sigma$, then the final justification that corresponds to this derivation is not warranted, since grounded (sceptical) argumentation semantics ensure that at least one of these literals will not be warranted in $\sigma$.

### 4.3.2 Warranted Plans

If plan projection for a plan $\pi$ succeeds, then $\pi$ is candidate plan. Following, the acceptability semantics we introduced for defeasible situation calculus, a warranted plan for a MPCP problem is a plan $\pi$ such that:

- There exists an acceptable argument for every goal literal in the top layer.
- For every action $A_k$ in $\pi$ applied in layer $k$ there exists an agent $i$, such that for

**Algorithm 11:** Labelling algorithm for plan projection graph

Plan projection graph $\mathcal{G}$;
Labelled plan projection graph $\mathcal{G}_L$;
$\mathcal{G}_L := \mathcal{G}$;
**foreach** *layer i starting from layer* 1 **do**
  **if** *l is even* **then** `// l is a literal layer`
    **foreach** *node n in current layer* **do**
      **if** *there exists a justification j leading to n labelled as W* **then**
        **if** *layer l does not contain the complement of n* **then**
          mark $n$ in layer $l$ as $W$;
        **else if** *every justification $j'$ leading to the complement of n is labelled as NW* **then**
          mark $n$ in layer $l$ as $W$;
        **else**
          $pref_n :=$ highest preferred justification connected to $n$ and labelled as $W$;
          $pref_{\overline{n}} :=$ highest preferred justification connected to $\overline{n}$ and labelled as $W$;
          **if** $pref_n > pref_{\overline{n}}$ **then**
            mark $n$ in layer $i$ as $W$;
          **else**
            mark $n$ in layer $l$ as $NW$;
      **else**
        mark $n$ in layer $l$ as $NW$;
  **else if** *l is odd* **then** `// i is a justification layer`
    **foreach** *node n in current layer* **do**
      **if** *every literal leading to n is labelled W* **then**
        mark $n$ in layer $l$ as $W$;
      **else**
        mark $n$ in layer $l$ as $NW$;
**return** $\mathcal{G}_L$;

every literal $L \in pre_i(A_k)$, there exists an acceptable argument with claim $L$ in layer $k$.

Similar to the DBAT plan acceptability semantics, a plan is warranted if its applicability and effectiveness is derived from the agents' collective beliefs, and the beliefs supporting it are "stronger" than any possible objections.

A special case of warranted plans are plans which raise no objections. We call these plans *undisputed plans*. A plan $\pi$ is an undisputed plan if it is a candidate plan and for every literal that is part of the support set of an argument claiming that the goal or a precondition of an action in the plan holds, its complement is not part of the relevant layer in the plan projection graph.

Planning is performed as a search for a sequence of actions that forms a warranted plan. Similar to the algorithms presented in the previous section, pruning the search is possible based on derivation and warrant information regarding action applicability.

The main difference between planning with DBATs and planning with MPCP problems with preference orderings over beliefs is a result of the use of default negation within successor state axioms in DBATs. As a result, defeasible derivations subsume derivations made using the state-transition function. Defeasible derivations can be made for arbitrary assumptions, even if there is strong indication that these do not hold. These assumptions are then disqualified in the argumentation phase. The MPCP reasoning mechanism does not rely on assumptions and results in deriving a reduced number of candidate plans. Accordingly, it reduces the necessary argument evaluation steps, since it essentially focuses on a smaller argumentation framework.

This benefit goes hand in hand with the requirement that the initial state is complete, since in order to be able to handle states with uncertainty in a similar fashion as in DBATs, a complicated specification of the state transition function is necessary. The state transition function is used numerous times within the search for candidate plans, and there is a significant advantage if it is quickly calculated. In addition, in order to effectively utilise standard planners for the construction of candidate plans, the specification of our planning domains should deviate as little as possible from the classical planning domain specification.

### 4.3.3 Planning using Classical Planners

The custom planners discussed in this chapter do not offer the level of code optimisation and fine-tuning of planning heuristics responsible for the efficiency of state-of-the-art planners. In this section, we take a different view toward the problem of identifying candidate plans, and focus on transforming the planning theory into a for-

mat which is suitable for standard planners.

The main differences between our set-theoretic planning formalism and the classical planning formalism are:

- absence of contradictory theories,
- multiple operator specifications,
- state transition function,
- preference values over beliefs, and
- derivation and Warrant relations.

Treatment of contradictions is performed through the use of literals instead of atoms. This is responsible for an increase in theory size, since every literal corresponds to two atoms. The state transition function is adapted from the standard specification in order to account for the relations among positive and negative literals referring to the same atom, when calculating what remains unaffected after the application of an action.

The task of finding warranted plans is strictly worse than planning. The reasons behind this are the extended size of the theory that is due to the use of literals and preference values, the additional calculations necessary for the computation of the state transition function, and the argumentation steps necessary to evaluate the warrant. The problem of finding candidate plans on the other hand can be viewed as a classical planning problem. This requires the construction of a planning theory that accounts for literal-based states, multiple specifications and the non-standard representation of the state transition function. The following section presents algorithms for planning with external classical planners, exploiting their highly optimised, heuristic search of the state space.

#### 4.3.3.1 Synthesising Candidate Plans

Contrary to classical planning, MPCP states can be *contradictory*, i.e. they may contain both positive and negative literals for the same atom. Standard planners usually represent states as sets of atoms. Atoms that are not part of a state correspond to negative literals. This representation enables a compact Boolean notation for every fluent. Every atom is either true or false in each state.

States have slightly different semantics in our system. In MPCP problems respecting the initial state completeness assumption, they describe what the agents have reason to believe in, rather than what the agents know. There are the following cases:

1. We have reasons to believe that the fluent takes a positive value in the state.
2. We have reasons to believe that the fluent takes a negative value in the state.
3. Both (1) and (2) are the case.

We introduce the set of symbols $\mathcal{L}_{\overline{p}} = \{\overline{q} \mid q \in \mathcal{L}_p\}$ to $\mathcal{L}$, and replace every negative ground and unground literal predicate $\neg q$ in the theory with an atom $\overline{q}$ from $\mathcal{L}_{\overline{p}}$, which has the same arity, constant and variable symbols.

The sets $\overline{Atoms}$ and $\overline{GroundAtoms}$ contain all unground and ground instances of the predicate symbols in $\mathcal{L}_{\overline{p}}$. After the application of the transformations every *state* $\sigma$ is a subset of $\overline{GroundAtoms}$. Also, sets containing preconditions of operators $pre(o)$ or conditions $\Gamma$ for a conditional effect are subsets of $\overline{Atoms}$, and effects $\phi$ are members of $\overline{Atoms}$.

The initial state is formulated according to the available literal information. If a positive literal $p$ is part of the initial state we add the atom $p$ to the new initial state. Accordingly, if we have reasons to believe that a negative literal $\neg p$ holds in the initial state, we add the atom $\overline{p}$.

Further transformations to the operator specification are necessary. First of all, we need to implicitly account for the removal of the complement of the effects of applied actions. This operation is usually performed by the planner's state transition function. The use of the auxiliary predicates conceals the semantic relation between atoms $p$ and $\overline{p}$ from the planner. As a result, the introduction of one of these as an effect of an action does not affect its complement.

In addition, the transformation needs to aggregate the operator specifications of multiple agents into a single specification. The operator schema may contain multiple triples of the form $\langle pre_i, o, eff_i \rangle$ regarding the same operator $o$, representing the specification of operator $o$ held by agent $i$. The transformation mechanism asserts that the constructed specification adheres to the following principles:

- Only the effects for applicable specifications must be triggered (preserving the link between preconditions and postconditions).
- Every specification implicitly describes which literals are not affected by the action (i.e. frame axiom).

Every specification of an operator defines a set of preconditions and a set of conditional effects. It is rational to assume that the effects are triggered if both the preconditions and the conditions of the conditional effect are satisfied by the state. In the aggregated operator specification, the preconditions of each specification also play the role of additional conditions to its conditional effects. For instance, if we have the specifications $\langle p, o, \{\langle c, e\rangle\}\rangle$ and $\langle p', o, \{\langle c', e\rangle\}\rangle$, the aggregated specification needs to contain the effects $\langle p \wedge c, e\rangle$ and $\langle p' \wedge c', e\rangle$. The precondition of the action is the disjunction of the preconditions of the individual specifications. For operator $o$, this would be $p \vee p'$.

Apart from describing the effects of actions, every operator specification implicitly describes which literals are not affected by it. Multiple specifications provide multiple implicit rules describing when literals are not affected by an action. Following the previous example, we focus on the specifications $\langle p, o, \{\langle c, e\rangle\}\rangle$ and $\langle p', o, \{\langle c', e\rangle\}\rangle$. According to the first specification, if we have reasons to believe that $p$ and $c$ hold in a state, then we have reasons to believe that $e$ holds in the resulting state after the application of the action. Therefore, according to this specification, we should not believe that $\overline{e}$ is the case in the next state. However, if we have reasons to believe that either $p'$ or $c'$ does not hold, then according to the second specification, $\overline{e}$ is not affected by the action. Consequently, if we have reasons to believe that it holds in the previous state, then $\overline{e}$ is also part of the resulting state.

Due to the use of auxiliary predicates, the planner is unaware of the conceptual relation between the fluents $p$ and $\overline{p}$. If this relation was transparent the planner would assert that the complements of the effects of the latest action do not hold. We achieve this manually by introducing additional effects. These effects act as frame rules and remove the complements (i.e. with respect to the $\overline{\ }$ symbol notation) of fluents that are added to the resulting state. In order to take the existence of multiple operator specifications and their implicit frame rules into account, we assert that these effects are only applicable when all specifications are in agreement regarding the relevant effect. If there exists at least one specification that describes reasons to believe that the literal is not affected, then we do not remove its complement.

Algorithm 12 describes the necessary process for pre-processing the planning problem. We slightly abuse the $\overline{\ }$ notation; $\overline{\psi}$ denotes the replacement of negated literals $\neg p$ with the corresponding new literals $\overline{p}$ in the negation normal form of the formula $\neg\psi$. The conditions of conditional effects here are considered to be expressions instead of sets of literals. A set of literals corresponds to the conjunction of all the literals in the set. Effects formulate the "add list" of operators, that is the literals that must be

**Algorithm 12:** Algorithm for the translation of the collective planning theory to a theory that can provide suitable input for a standard off the self planner

Replace every negated literal $\neg p$ with a new literal $\overline{p}$;
Create $S', O', \sigma'_0$ and $g'$;
**foreach** *operator* $o$ **do**
  $effect := \emptyset$;
  **foreach** *fluent literal* $e$ *such that* $\exists \langle \phi, e \rangle \in eff(o)$ **do**
    **foreach** $\langle pre, o, eff \rangle \in O'$ **do**
      $\psi_e := \psi_e \vee pre(o) \wedge (\bigwedge_{\langle \phi, e \rangle \in eff(o)} \phi)$;
      **if** $e \notin eff(o)$ **then** $\psi_{\overline{e}} := \psi_{\overline{e}} \vee pre(o)$;
      **else** $\psi_{\overline{e}} := \psi_{\overline{e}} \vee (\overline{pre(o)} \vee (\bigvee_{\langle \phi, e \rangle \in eff(o)} \overline{\phi})$;
    $effect := effect \cup \langle \psi_e, e \rangle$;
    $effect := effect \cup \langle \overline{e} \wedge \psi_e \wedge \neg \psi_{\overline{e}}, \neg \overline{e} \rangle$;
  $precondition := \bigvee_{\langle pre, o, eff \rangle \in O'} pre(o)$;
  $O'' := O'' \cup \{\langle precondition, o, effect \rangle\}$;
$\Sigma' := (S', O'', \gamma_{classical})$;
$P' := \langle \Sigma', \sigma'_0, g' \rangle$;
**return** $P'$;

added to a state resulting from the application of the operator. Equivalently, the complements of effects correspond to the "delete list", that is the list of literals that are deleted after the application of the operator. The symbol $\gamma_{classical}$ denotes the standard state transition function used in classical planning.

The planning problem resulting from the application of the transformation is called a *candidate planning problem*, and is the triple $P' = \langle \Sigma', \sigma'_0, \gamma_{classical} \rangle$. Every solution to the candidate defeasible planning problem is a candidate plan to the corresponding multiagent defeasible planning problem. This holds because the two formalisms produce equivalent state-based results, with the auxiliary literals in $P'$ corresponding to negative literals in $P$. The initial state $\sigma'_0$ is constructed so that the states are equivalent with respect to this principle. In any successor state, if a literal is added by $\gamma$ due to an applicable conditional effect, then the atom corresponding to this literal is added by $\gamma_{classical}$, since an equivalent effect is added to $O'$. Additional effect rules introduced in $O'$ assert that literals introduced by $\gamma$ due to inertia are also introduced by $\gamma_{classical}$.

The translation algorithm we described in the previous section provides a suitable

theory to be input to a standard off-the-shelf planner. After parsing the theory, the external planner searches the state-space for a candidate plan. If one is returned, its warrant state can be evaluated accordingly.

Most planners are designed to return a single plan. However, if this plan is not warranted, additional search is necessary. By performing suitable modifications to the planner's input, we must guide the planer to provide a different candidate plan. Ideally, the planner should utilise the information obtained during the argumentation phase regarding the reasons responsible for rejecting the previously synthesised candidate plans. In this way, there would be a guarantee that future plans do not fail for the same reasons.

#### 4.3.3.2 Iterative Revision-Based Planner

Algorithm 13 describes the Iterative Revision-Based (IRB) planner which iteratively calls an external, off-the-shelf planner, evaluates whether the returned candidate plan is warranted, and revises the theory so that future plans returned from the external planner do not suffer from the same contradictions. Similar to Algorithm 7, the search is pruned for plans known to be unwarranted and the argumentation process is limited to candidate plans. The IRB planning procedure resolves contradictions that are related to candidate plans. This is particularly helpful in large domains containing information irrelevant to the goal.

Revising initial state beliefs and simple effects of specifications with the same pre-conditions is straightforward. We select the one with the highest preference. However, this is not the case for contradictions in operator specifications. For example, consider two specifications of an operator, one with the conditional effect $\langle\{Power(l)\}, Light(l)\rangle$, and the other with $\langle\{Was\_Sunny(l)\}, Light(l)\rangle$. Also assume that no other effect in the specifications is related to the predicate $Light(l)$.

Assume that there are no other effects causing $Light(l)$ or $\neg Light(l)$, and that there are no reasons to believe that $l$ is lit before applying the operator. According to the first specification, we have reason to believe that $l$ is lit only if $l$ is connected to a power source, whereas according to the second this is the case only if the sun is shining.

The two specifications lead to contradictory derivations only in situations in which we have reasons to believe exactly one of the statements $Power(l)$ and $Was\_Sunny(l)$. The resolution of such contradictions depends both on our preferences over the specifications, and on our preferences over conditions $Power(l)$ and $Was\_Sunny(l)$ in the previous state. In the general case, resolution of contradictions is situation-dependent

**Algorithm 13:** Iterative Revision-Based Planner

$NW = \emptyset$;
$P' := P$;
**repeat**
  $\pi := call\_external\_planner(P')$;
  **if** $project(\pi)$ **then return** $\pi$**;**
  **else**
    $NW := NW \cup not\_warrated(P', \pi)$**;**
    $P' := revise(NW, P)$**;**
**until** *no new plans exist*;

(or plan-dependent), and encoding this information in the planning operators requires the encoding of action histories in the state space of the planning problem. This is obviously not practical and would lead to combinatorial explosion.

To overcome this issue, we follow a heuristic approach: We modify the operator responsible for literals that are not warranted in future situations, and caused previous plans to fail, so that these ground literals are not produced by the operator. This affects only the specific ground literal and not the other instances that could arise due to the same effect clause. This method is not complete, since by deleting the problematic ground effect we make the generalisation that this effect always leads to contradictory beliefs, which will not always be the case. However, soundness is preserved as the warrant status of returned plans are evaluated externally of the employed planner.

#### 4.3.3.3 GHC Planner

Algorithm 14 presents our GHC planner. This planner operates under a similar principle with the EHC algorithm presented in Section 4.2.2.5. GHC selects from the literals that are warranted in the initial state and calls an external planner for a candidate plan. If a plan is returned, it greedily traverses through the actions of the plan to the successor states, until all actions have been applied or an action whose application is not warranted in the corresponding state is met. The greedy approach allows minimising calls to the external planner, enabling the quick traversal to states which can potentially achieve the goal.

When the application of the actions finishes, if the goal is warranted, the planner returns this plan. Alternatively, the planner searches for a state with a better quality

**Algorithm 14:** GHC Planner

$\sigma := I$;
*history* $:= \langle\rangle$;
$\sigma_w := \{L \mid L \in \sigma$ and $L$ is warranted$\}$;
*candidate* $:=$ *call_external_planner*$(\sigma_w)$;
*candidate*$_w :=$ the maximal sequence of actions in *candidate* which are warranted in sequence w.r.t. $\sigma$;
$\sigma := \gamma(candidate_w, \sigma)$;
*history* $:=$ *history* $+$ *candidate*$_w$;
**while** *a threshold* $\varepsilon$ *is not reached* **do**
  *progressed* $:=$ false;
  **if** *all goal conditions are warranted in* $\sigma$ **then**
    **return** *history*;
  **foreach** *action A whose application is warranted in* $\sigma$ **do**
    $\sigma' := \gamma(A, \sigma)$;
    $\sigma'_w := \{L \mid L \in \sigma'$ and $L$ is warranted$\}$;
    **if** *all goal conditions are warranted in* $\sigma'$ **then**
      **return** *history* $+ \langle A \rangle$;
    *candidate*$' :=$ *call_external_planner*$(\sigma'_w)$;
    *candidate*$'_w :=$ the maximal sequence of actions in *candidate*$'$ which are warranted in sequence w.r.t. $\sigma'$;
    **if** $|candidate'_w| > 0$ **then**
      $\sigma := \gamma(candidate'_w, \sigma')$;
      *history* $:=$ *history* $+ \langle A \rangle +$ *candidate*$'_w$;
      *progressed* $:=$ *true*;
      *break*;
  **if** *progressed* $=$ *false* **then**
    **return** *null*;
**return** *null*;

in the neighbourhood of the current state, i.e. the states that can be reached with one transition by an action whose application is warranted. The quality of the state is calculated as the number of actions in the candidate plan the external process returns for this state, whose application is warranted in sequence. The quality of the current

state is 0. Higher numbers correspond to better heuristic values. If a state with a higher heuristic value is found, then the planner greedily selects it and repeats the process with this state as the current state. The greedy approach allows the planner to quickly move to better states without searching through the entire neighbourhood, which in extensive domains may include thousands of states. This is very important as in every step the planner is required to performed argumentation steps in combination with calls to the external planner.

In order to minimise the possibility of circles in neighbouring states, the order actions are considered is non-deterministically selected in every iteration. The standard way to avoid such cases is to maintain lists of the traversed states. However, this is not effective in this case, since because states containing the same literals are not necessarily equivalent, since they may entail different warrant results. Hence, in order to comprehensively evaluate whether two states are equivalent we must compare their sets of literals, their warrant status and their relevant preference orders. The alternative is to evaluate state equivalence based on the history of actions that led the planner to these states, i.e. in a similar fashion to a situation term in DBATs.

GHC evaluates the warrant status of literals based on a forward and a backward step. The literals that are contained in a state is computed in a forward manner. Subsequently, we evaluate their warrant status based on a labelling process in a backwards chaining manner. If the warrant results of the predecessor states have been already evaluated before we traversed to the current state, these results are reused. We store the warrant information and (for warranted literals and justifications) the preference value. Accordingly, if the warrant evaluation results regarding the predecessor state have been calculated, the labelling process is limited to one backward step.

In order to increase the potential of the returned candidate plans, GHC evaluates the warrant status of the literals in the current state, and feeds the external planner with a state that does not include literals that are not warranted. As a result, the external planner solves a candidate planning problem that does not necessarily respect the initial state completeness assumption. This does not affect the correctness of the GHC algorithm, since the returned candidate plans are used to calculate the heuristic value of a state, which only indicates the direction of the search. The actual state transitions are performed based on complete states.

Heuristic search may lead to a local maximum, or continue without reaching a goal state. In order to ensure termination we limit the path the planner can traverse. Additionally, in order to safeguard termination in domains with extensive size of actions,

we bound the number of states that can be considered.

## 4.4 Summary

This chapter investigates the problem of synthesising warranted plans that solve planning problems in which agents share goals, but hold different views on the initial state and the operator specifications. A naive implementation of this process based on propositional argumentation is impractical due to the size of the generated ground theories. To tackle this problem, we propose a series of planning algorithms that are based on specific queries for the derivation and evaluation of their warrant status of literals in the situation resulting after the application of relevant actions. With this, we manage to prune the search space. In order to further increase efficiency, we present strategies for selecting the most prominent point of attack during argumentation. In addition, taking inspiration from the planning literature, we present a planning heuristic based on the defeasible derivation relation that can be used to guide the search and prioritise actions.

Apart from the presented algorithms that are based on DBAT, this chapter focuses on the set-theoretic MPCP representation and specifies arguments, acceptability and warranted plans. With this formalism, we focus on the differences between MPCP and classical planing. Based on these observations, we propose algorithms that delegate the search for candidate plans to efficient external planners.

Contrary to standard state space planning, the search for candidate plans, is bound to the notion of situation (or history), since the warrant status of literals is relevant to the history of actions that led to this state. States in which the same literals can be derived do not always share the same warrant results. Warrant information can be introduced within the state. However, this results in an significant increase in the state space.

The main contributions of this chapter are the following:

- Focus on the algorithmic aspects of MPCP.
- Formulation of the planning sub-problem of MPCP in a format suitable for classical planners.
- Presentation of pruning strategies, heuristics, and algorithms for planning with DBATs and MPCP problems.

# Chapter 5

# Dialogue Protocols

## 5.1 Introduction

The methods described in the previous chapter provide centralised solutions to the problem of multi-perspective cooperative planning. Such techniques require communication of all beliefs prior to planning. In the general case, this is not optimal since it involves the communication of beliefs which neither support nor object to potential plans. Moreover, in special cases, this process may be potentially problematic. For example, agents with privacy constraints may not want to subscribe to mechanisms that require them to share their entire knowledge base, including beliefs that do not provide insight to the problem at hand.

This chapter presents a family of dialogue-based protocols for the distribution of the solution finding mechanism. The dialogue-based approach allows cooperative agents to search the space of potential plan proposals, resolve contradictory beliefs and reach agreement while aligning their knowledge. This is achieved by exchanging meaningful arguments regarding concrete potential proposals.

The approach presented in this chapter is based on the combination of argumentation theory and (defeasible) situation calculus in a distributed setting. The dialogue-based nature of the mechanism enables the distribution of the argumentation process. Agents initiate discussions about concrete plans they have generated individually. They then collaboratively resolve contradictions in their beliefs that are relevant to the evaluation of the discussed plans. As a result, this process avoids the communication of beliefs that are irrelevant to the specific problem and the resolution of irrelevant inconsistencies.

We present an abstract argument-based protocol that enables discussion of can-

didate proposals and extend it for the specific problem of arguing about plans. The dialogue is broken down into sub-dialogues, which discuss alternative proposals. If a sub-dialogue fails, the protocol ensures that the source of the disagreement is discovered and resolved, and that the knowledge of the agents is gradually aligned as participants' local misconceptions are uncovered. The main dialogue-based protocol is extended to enable multi-party dialogue and collaborative argument generation.

The work presented in this chapter has been previously published by Belesiotis et al. (2009) and Belesiotis et al. (2010).

## 5.2 Iterated Disputes

We start by describing *iterated disputes*, our two-agent dialogue framework at the level of abstract argumentation (Dung, 1995), together with a protocol for iterated argumentation that is suitable for arguing over plan proposals as we will later show. We assume two-player situations; in the case of more than two agents, our results carry over assuming pairwise dialogues are conducted between all agents to reach agreement among the full set of agents.

Argumentation theory provides strong theoretical foundations for formally defining the notion of acceptability, and mechanisms for the identification and resolution of conflicts. An abstract argumentation approach provides modular representation and abstraction between different aspects of the problem. Argumentation-based dialogue enables the agents to share their beliefs together with justifications explaining how the knowledge has been obtained. Justifying claims enables the resolution of conflicts, because it provides additional information regarding the reasons why beliefs should be accepted.

### 5.2.1 Dialogue Protocol

Figure 5.1 outlines the *iterated disputes* dialogue protocol, which extends *two-party immediate response disputes* for grounded argumentation semantics (Dunne and Bench-Capon, 2003), in order to allow the evaluation of multiple proposals. Every iteration is followed by an argument revision step which aligns the argument sets of the agents. We consider the agents to have distinct argument sets, instead of sharing the same pool of arguments, which is usually the case in disputes.

The agents evaluate the acceptability of a proposal through a dispute (Dunne and

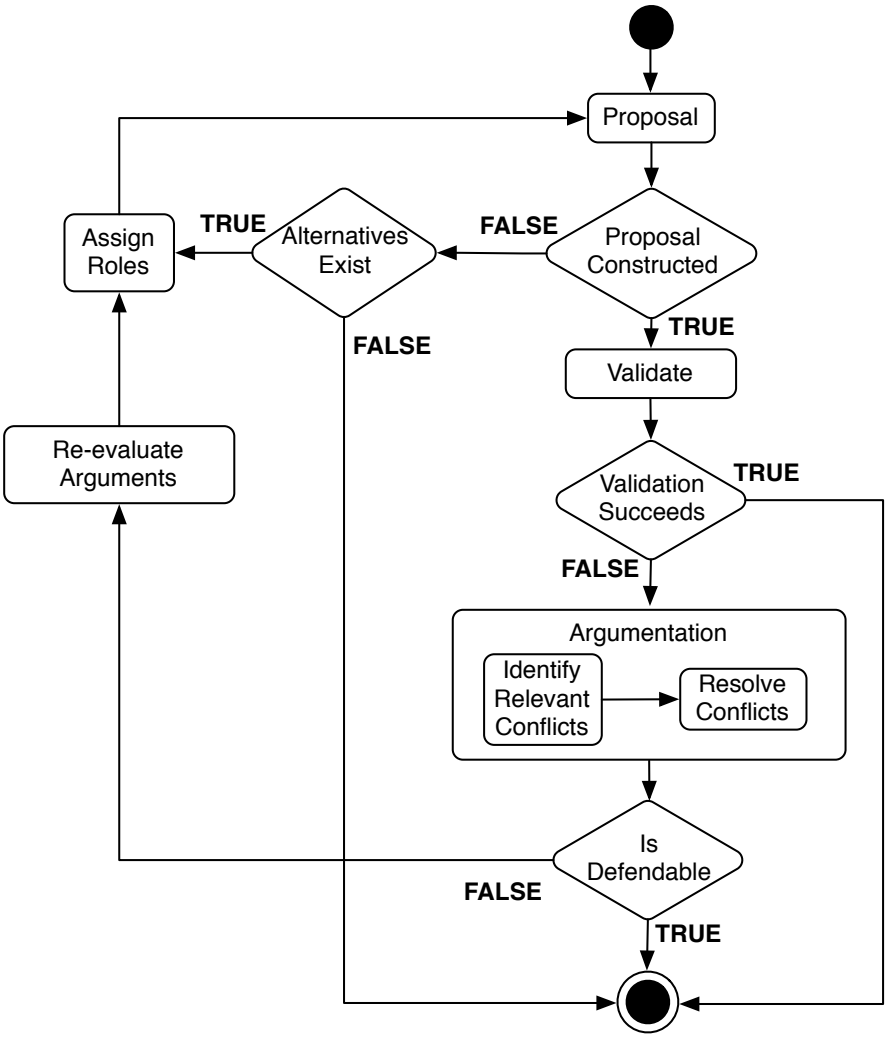



Figure 5.1: Outline of an iterated dispute

Bench-Capon, 2003). The agent that made the proposal plays the role of the proponent *PRO*, leaving the role of opponent *OPP* to the other party. The proponent is responsible for constructing arguments in favour of the proposal, while the opponent's role is to show that the proposal should not be accepted. The dialogue game progresses with each agent presenting arguments defeating the arguments of their rival. Iterated disputes facilitate the discussion of different proposals in sequence.

The following analysis follows Dunne and Bench-Capon (2003). Let an abstract argumentation framework $AF = \langle Args, Defs \rangle$, with $Args = Args_{PRO} \cup Args_{OPP}$ the union of the arguments that are available to the proponent and the opponent respectively. A *dispute tree for some argument* $\rho$ *in Args*, denoted by $\mathcal{T}$, is a tree with root $\rho$ whose vertices and edges are subsets of *Args* and *Defs*, respectively. The edges in a dispute

tree are directed from vertices to their parent node. $Depth(\mathcal{T},\alpha)$ for an argument $\alpha$ in a dispute tree $\mathcal{T}$ denotes the number of edges in the dispute line from $\alpha$ to the root of the tree. $Children(\mathcal{T},\nu,e)$ represents all arguments in $\mathcal{T}$ that have $\nu$, which is located in depth $e$, as their parent node.

A *dispute line* is a path in the dispute tree:

$$\nu_k \rightarrow \ldots \rightarrow \nu_1 \rightarrow \nu_0 = \rho\,.$$

For every two consecutive arguments $\nu_j$ and $\nu_{j-1}$ in a dispute line it holds that $\nu_j$ defeats $\nu_{j-1}$ (i.e. $\nu_j \rightarrow \nu_{j-1}$).

A dispute line is called *open/closed* if the agent who has to make the following move is able/unable to defeat the other party's most recent move. A closed dispute line for some argument $\rho$ is a *failing defence/attack of P* if the leaf node argument move has been made by the opponent/proponent.

A *dispute* for some argument $\rho$ is a sequence of *moves*:

$$d = \langle m_1, m_2, \ldots, m_k, \ldots \rangle$$

affecting a dispute tree that has $\rho$ as its root. Every dispute evaluates the acceptability of a candidate proposal. Dispute $d$ after $k$ moves will be denoted as $d_k$. The state of the dispute $d_k$ is a tuple:

$$State(d_k) = \langle \mathcal{T}_k, \nu_k, CS_k^{PRO}, P_k, Q_k, Args_k \rangle\,.$$

$\mathcal{T}_k$ is the dispute tree after the most recent argument move $\nu_k$. $CS_k^{PRO}$ contains arguments that have been presented by the proponent in the current dispute line providing defence on $\rho$. $P_k$ is the set of arguments the proponent has presented in the current dispute tree. $Q_k$ contains sets of arguments presented by the proponent that failed to defend $\rho$. $Args_k$ represents the arguments that have been exchanged in the dispute by both agents, excluding the proposal argument. We will refer to a dispute as being *closed* if the agent who has to make the following move is unable to make a move affecting the tree of the dispute. A closed dispute line for some argument $\rho$ is a *failing defence/attack* if the final move affecting the dispute tree was made by the opponent/proponent.

An *iterated dispute* is a sequence of disputes $\mathbb{d} = \langle d_1, d_2, \ldots, d_l, \ldots \rangle$. An iterated dispute can be rewritten as a sequence of *legal* moves $\mathbb{d} = \langle m_{1,0}, m_{1,1}, \ldots, m_{l,k}, \ldots \rangle$, where $m_{l,k}$ denotes the $k$th move of the $l$th dispute. The boolean function $Legal(m,\mathbb{d})$ succeeds if all the *conditions* specified by the move hold before the move is applied.

$\mathit{CurrentDispute}(\mathbb{d})$ returns the most recent dispute, $\mathit{PreviousMoveType}(\mathbb{d})$ denotes the type of the most recent move, and $\mathit{Proposal}(d)$ is $d$'s root argument.

The applicability of dialogue moves is specified by sets of conditions and effects. The following moves dictate the rules that achieve grounded semantics. We extend the moves proposed by (Dunne and Bench-Capon, 2003) by introducing additional ones to allow dialogue over multiple disputes and additional structures to maintain unsuccessful proposals. The *propose* move initiates a new dispute. The agents are restricted to propose new arguments from $\mathcal{P}$, which is the set of all possible proposals.

| $m_{l,0} = \langle \mathit{propose}, i, \rho \rangle$ | |
|---|---|
| Conditions: | Effects: |
| $\mathit{PreviousMoveType}(\mathbb{d}) \in \{\mathit{close}, \mathit{no\text{-}proposal}\}$, or $\mathbb{d} = \langle\rangle$ | Roles are switched |
| | $\mathcal{T}_{l,0} := \langle \rho \rangle$ |
| $\forall d$ in $\mathbb{d}$, $\mathit{Proposal}(d) \neq \rho$ | $\nu_{l,0} := \rho$ |
| $\rho \in GE_{\langle Args^i_{l-1} \cup \{P\}, Defs \rangle}$ | $CS^{PRO}_{l,0} := \{\rho\}$ |
| $\rho \in \mathcal{P}$ | $P_{l,0} := \{\rho\}$ |
| | $Q_{l,0} := \emptyset$ |
| | $Args_{l,0} := \emptyset$ |

The *no-proposal* move is made when an agent is unable to present a new proposal.

| $m_{l,0} = \langle \mathit{no\text{-}proposal}, i \rangle$ |
|---|
| Conditions: |
| $\mathit{PreviousMoveType}(\mathbb{d}) = \mathit{close}$ or $\mathbb{d} = \langle\rangle$ |
| $\nexists \rho$ *s.t.* $\mathit{Legal}(m, \mathbb{d})$, for $m = \langle \mathit{propose}, i, \rho \rangle$ |

The *terminate* move can be used after a no-proposal move in order to terminate the discussion when no alternative proposals can be presented by either one of the agents. In this case, the iterated dispute finishes without reaching a mutually acceptable solution.

| $m_{l,0} = \langle \mathit{terminate}, i \rangle$ |
|---|
| Conditions: |
| $\mathit{PreviousMoveType}(\mathbb{d}) = \mathit{no\text{-}proposal}$ |
| $\nexists \rho$ *s.t.* $\mathit{Legal}(m, \mathbb{d})$, for $m = \langle \mathit{propose}, i, \rho \rangle$ |

*Counter* moves respond to the other party's most recent argument in a dispute. Sequences of counter moves expand the current dispute tree in a depth-first manner. In order to conform to grounded argumentation semantics, the proponent is not allowed to present conflicting arguments, or repeat the same arguments in the same dispute line. In addition, the proponent is prohibited from repeating lines of defence that have already failed in the current line of defence. This is achieved by disabling *PRO* to formulate a line of defence extending a line that has been added in $Q_{l,k}$.

| $m_{l,k} = \langle counter, OPP, \upsilon \rangle$ | |
|---|---|
| Conditions: | Effects: |
| $PreviousMoveType(\mathbb{d}) \in$ $\{propose, counter, retract\}$ | $T_{l,k} := T_{k-1} + \langle \upsilon, \nu_{k-1} \rangle$ |
| $\upsilon \in Args^{OPP}_{l-1} \cup CS^{PRO}_{l,k-1}$ | $\nu_{l,k} := \upsilon$ |
| $\upsilon \rightarrow \nu_{l,k-1}$ | $CS^{PRO}_{l,k} := CS^{PRO}_{l,k-1}$ |
| | $P_{l,k} := P_{l,k-1}$ |
| | $Q_{l,k} := Q_{l,k-1}$ |
| | $Args_{l,k} := Args_{l,k-1} \cup \{\upsilon\}$ |

| $m_{l,k} = \langle counter, PRO, \upsilon \rangle$ | |
|---|---|
| Conditions: | Effects: |
| $PreviousMoveType(\mathbb{d}) \in$ $\{counter, backup\}$ | $T_{l,k} := T_{k-1} + \langle \upsilon, \nu_{k-1} \rangle$ |
| $\upsilon \in Args^{PRO}_{l-1}$ | $\nu_{l,k} := \upsilon$ |
| $\upsilon \rightarrow \nu_{l,k-1}$ | $CS^{PRO}_{l,k} := CS^{PRO}_{l,k-1} \cup \{\upsilon\}$ |
| $\upsilon \notin CS^{PRO}_{l,k-1}$ | $P_{l,k} := P_{l,k-1} \cup \{\upsilon\}$ |
| $CS^{PRO}_{l,k-1} \cup \{\upsilon\}$ is conflict-free | $Q_{l,k} := Q_{l,k-1}$ |
| $\forall R \in Q_{l,k},\ R \not\subseteq P_{l,k-1} \cup \{\upsilon\}$ | $Args_{l,k} := Args_{l,k-1} \cup \{\upsilon\}$ |

The notation $T_{k-1} + \langle \upsilon, \nu_{k-1} \rangle$ represents the dispute tree resulting from adding the node $\upsilon$ and the edge $\langle \upsilon, \nu_{k-1} \rangle$ to the dispute tree $T_{k-1}$.

If *PRO*'s most recent argument cannot be countered, *OPP* makes an alternative defeat using the *backup* move. In this way, the opponent is allowed to backtrack and focus on the most recent argument in the current dispute tree that was presented by *PRO* and is defeated by an alternative argument that is held by *OPP*.

| $m_{l,k} = \langle backup, OPP, \upsilon, \chi, e \rangle$ |
|---|
| Conditions: |
| $PreviousMoveType(\mathbb{d}) \in \{counter\}$ |
| $\upsilon \in Args^{OPP}_{l-1} \cup CS^{PRO}_{l,k-1}$ |
| $\chi = \nu_b$ is the most recent argument in the dispute line |
| $\nu_n \rightarrow \ldots \rightarrow \nu_b \rightarrow \ldots \rightarrow \rho$ for which: |
| $-\ \delta = Depth(\chi) + 1$ is odd |
| $-\ \upsilon \rightarrow \chi$ |
| $-\ \upsilon \notin Children(\mathcal{T}_{l,k}, \chi, Depth(\chi))$. |
| Effects: |
| $\mathcal{T}_{l,k+1} := \mathcal{T}_k + \langle \upsilon, \chi \rangle$ |
| $\nu_{l,k+1} := \upsilon$ |
| $CS^{PRO}_{l,k+1} := CS^{PRO}_{l,k-1} \setminus \{\nu_n, \nu_{n-2}, \ldots, \nu_{b+1}\}$ |
| $P_{l,k} := P_{l,k-1}$ |
| $Q_{l,k} := Q_{l,k-1}$ |
| $Args_{l,0} := \emptyset$ |

The *retract* move can be used by the proponent in order to attempt to provide an alternative line of defence if it is not possible to counter an argument presented by *OPP*. In order to ensure that the proponent does not repeat the same line of defence, as it has been already shown to fail, the proponent's arguments are stored in $Q_{l,k}$.

| $m_{l,k} = \langle retract, PRO \rangle$ | |
|---|---|
| Conditions: | Effects: |
| $PreviousMoveType(\mathbb{d})$ | $\mathcal{T}_{l,k+1} := \langle \rho \rangle$ |
| $\in \{counter, backup\}$ | $\nu_{l,k+1} := \rho$ |
| $\nexists \chi\ s.t.\ Legal(m, \mathbb{d})$ for | $CS^{PRO}_{l,k+1} := CS^{PRO}_{l,0}$ |
| $m = \langle counter, PRO, \chi \rangle$ | $P_{l,k} := P_{l,0}$ |
| | $Q_{l,k} := Q_{l,k-1} \cup \{P_{l,k-1}\}$ |
| | $Args_{l,k} := Args_{l,k-1}$ |

The *accept proposal* move is preformed by the opponent, closing the most recent dispute as a *failing attack* and terminating the dialogue in favour of the most recent proposal.

| $m_{l,k+1} = \langle accept, OPP \rangle$ |
|---|
| Conditions: |
| $PreviousMoveType(\mathfrak{d}) \in \{propose, counter\}$ |
| $\nexists \mathfrak{v}$ *s.t.* $Legal(m, \mathfrak{d})$, for $m = \langle counter, OPP, \mathfrak{v} \rangle$ |
| $\nexists \mathfrak{v}, \chi, e$ *s.t.* $Legal(m, \mathfrak{d})$ for $m = \langle backup, OPP, \mathfrak{v}, \chi, e \rangle$ |
| Effects: |
| $Args_l^{PRO} := Args_{l-1}^{PRO} \cup Args_{l,k}$ |
| $Args_l^{OPP} := Args_{l-1}^{OPP} \cup Args_{l,k}$ |
| $Proposal(d_l)$ is accepted |

The *close dispute* move is available to the proponent and closes the most recent dispute as a *failing defence*.

| $m_{l,k+1} = \langle close, PRO \rangle$ |
|---|
| Conditions: |
| $PreviousMoveType(\mathfrak{d}) \in \{counter, backup\}$ |
| $\nexists \mathfrak{v}$ *s.t.* $Legal(m, \mathfrak{d})$, for $m = \langle counter, PRO, \mathfrak{v} \rangle$ |
| $\nexists \mathfrak{v}, \chi, e$ *s.t.* $Legal(m, \mathfrak{d})$, for $m = \langle retract, PRO, \mathfrak{v}, \chi, e \rangle$ |
| Effects: |
| $Args_l^{PRO} := Args_{l-1}^{PRO} \cup Args_{l,k}$ |
| $Args_l^{OPP} := Args_{l-1}^{OPP} \cup Args_{l,k}$ |

The restrictions specified by the protocol are quite liberal, since they do not always impose a singe move. For instance, an agent that holds different defeaters to the most recent argument presented by the other party can make multiple legal *counter* moves. In order to automate the move selection process, and ensure that the dialogue leads to correct results, we pair the dialogue protocol with the *confident strategy*. A *strategy* is a set of rules which select exactly one move from the set of all legal moves. The confident strategy constructs a move based on a complete ordering over all possible legal options. Preference over moves is calculated according to:

1. The following ordering over move types:

$$\langle counter, backup, retract, close, accept, propose, no\text{-}proposal, terminate \rangle.$$

2. The preference level of the argument presented by the move, for moves of the same type.

If the preference ordering over arguments is partial, a complete ordering is obtained by prioritising moves that are equally preferred in a non-deterministic manner.

The confident strategy ensures that every relevant argument is exchanged eventually. In addition to this, the strategy expands the argumentation tree in a depth-first manner, which allows us to prune paths that do not alter the acceptability status of the proposal argument that is being evaluated.

### 5.2.2 Properties

This section presents important properties of the abstract argumentation protocol of iterated disputes. The following proofs assume that the dialogue is conducted between two agents following confident strategies.

**Proposition 18.** *An iterated dispute for agents with finite argument sets always terminates.*

*Proof.* The agents' arguments are finite. The proponent cannot repeat the same arguments in the same dispute line, and cannot repeat infinite *backup* moves as each one represents an alternative line of defence. Therefore, disputes always terminate. If there exists a dispute that is a failing attack of the proposal, the iterated dispute will terminate. We show that there can be no infinite sequence of disputes that are all failing defences. For proposal $\rho$ and dispute $l+1$, if $d_{l+1}$ is a failing defence of $\rho$, there exists a set of arguments *OPP* against which $\rho$ cannot be defended. *PRO* can only present proposals that are part of $GE_{\langle Args_l^{PRO} \cup \{\rho\}, Defs\rangle}$, which are defended against all defeats from $Args_l^{PRO}$. Since $\rho$ is not defended against all defeats in the dispute, there exists at least one argument $\beta$ that was presented by *OPP* and is not part of $Args_l^{PRO}$. The agents have finite argument sets and after every dispute they learn the arguments presented by the other party. So there cannot be an infinite sequence of disputes that are failing attacks. Therefore, an iterated dispute always terminates. □

In order to prove soundness, we introduce two key lemmas. The following lemma shows that, if a dispute $d_l$ is a failing attack of a proposal $\rho$, then $\rho$ will be in the grounded extension of the argumentation framework $\langle Args_l^{OPP} \cup \{\rho\}, Defs\rangle$, where $Args_l^{OPP}$ are the arguments that the opponent will know after the dispute terminates.

**Lemma 1.** *Let $d_l$ be a closed dispute about $\rho$ between agents PRO and OPP, with finite argument sets $Args_0^{PRO} = \mathcal{A}$ and $Args_0^{OPP} = \mathcal{B}$ that follow a confident strategy. If $d_l$ is a failing attack of $\rho$ then $\rho \in GE_{AF_l^{OPP}}$ for the argumentation framework*

$AF_l^{OPP} = \langle Args_l^{OPP} \cup \{\rho\}, Defs\rangle$, *where* $Args_l^{OPP}$ *denotes the arguments known to OPP after dispute* $d_l$.

*Proof.* All nodes presented by *OPP* in the dispute tree $\mathcal{T} = \mathcal{T}_{l,k}$ have one child node presented by *PRO*, since the proponent can only add a new argument to the dispute tree using the counter move. A dispute closes as a failing attack if the proponent counters every counter and backup move made by the opponent. Therefore, all leaf nodes are presented by *PRO*. We will show that all arguments of even depth in $\mathcal{T}$ are part of the grounded extension of $AF_l^{OPP}$ by induction over the distance between them and the leaf nodes in the tree.

(Base Case) For $distance = 0$, let $\mathcal{V}_0$ be the leaf node arguments of depth $n$. These arguments were presented by *PRO*. Also, $\forall \alpha_n \in \mathcal{V}_0, \nexists \beta_{n-1} \in Args_{OPP} \cup CS_{PRO}$ such that $\beta_{n-1} \rightarrow \alpha_n$, because if such an argument existed *OPP* would have presented it, due to the specification of the confident strategy and the counter and backup moves. Let $\mathcal{F}_{OPP}^i$ be the characteristic function of $AF_l^{OPP}$. All leaf node arguments in $\mathcal{V}_0$ are part of $\mathcal{F}_{OPP}^1$, since there are no arguments in $Args_l^{OPP}$ defeating them. So $\mathcal{V}_0 \subseteq GE_{AF_l^{OPP}}$.

(Induction Step) We assume that the property holds for arguments of distance $k$ from the leaf nodes, $\mathcal{V}_k \subseteq GE_{AF_l^{OPP}}$. We will show that the property holds for arguments of distance $k+2$. All arguments of distance $k+2$ from the leaf node $\mathcal{V}_{k+2}$ are defeated by an argument in $\mathcal{V}_{k+1}$, which are in turn defeated by arguments in $\mathcal{V}_k$. Also, there is no other $\beta'_{k+1} \in Args_l^{OPP}$ that defeats any argument in $\mathcal{V}_{k+2}$ that has not been presented, because of *OPP*'s strategy. According to the induction step $\mathcal{V}_k \subseteq GE_{AF_{OPP}}$, so $\mathcal{V}_{k+2} \subseteq GE_{AF_l^{OPP}}$.

Therefore, all arguments of even distance from the leaf nodes will be part of $GE_{AF_l^{OPP}}$, including $\rho$.

□

The following lemma shows that for agents with finite and conflict-free initial argument sets, which have exchanged subsets of their arguments, if a proposal is acceptable with respect to the arguments both agents know, then it will be also acceptable with respect to the union of both agents' arguments.

**Lemma 2.** *Let* $AF_1 = \langle \{\rho\} \cup \mathcal{A} \cup \mathcal{B}', Defs\rangle$, $AF_2 = \langle \{\rho\} \cup \mathcal{A}' \cup \mathcal{B}, Defs\rangle$ *and* $AF = \langle \{\rho\} \cup \mathcal{A} \cup \mathcal{B}, Defs\rangle$, *with* $\mathcal{A}$, $\mathcal{B}$ *finite, conflict-free argument sets,* $\mathcal{A}' \subseteq \mathcal{A}$ *and* $\mathcal{B}' \subseteq \mathcal{B}$. *If* $\rho \in GE_{AF_1} \cap GE_{AF_2}$ *then* $\rho \in GE_{AF}$.

*Proof.* We will first show by induction on the characteristic function $\mathcal{F}_{AF_1}$ that $\forall \beta \in \mathcal{B}'$ *s.t.* $\beta \in GE_{AF_1}$ it holds that $\beta \in GE_{AF}$. It holds that $\mathcal{B}$ is conflict-free, and $\beta, \rho \in GE_{AF_1}$, therefore all arguments defeating $\beta$ will be in $\mathcal{A}$.
(Base case) $\forall \beta \in \mathcal{F}^1_{AF_1}$, there exists no argument defeating $\beta$ in $\{\rho\} \cup \mathcal{A} \cup \mathcal{B}'$. There will be no argument defeating $\beta$ in $\{\rho\} \cup \mathcal{A} \cup \mathcal{B}$, so $\beta \in GE_{AF}$.
(Induction step) We assume that $\forall \beta \in \mathcal{F}_{AF_k}$, $\beta \in GE_{AF}$, and we show that is holds for $\forall \beta \in \mathcal{F}_{AF_{k+1}}$. All arguments defeating $\beta$ in $\{\rho\} \cup \mathcal{A} \cup \mathcal{B}'$ are also part of $\{\rho\} \cup \mathcal{A} \cup \mathcal{B}$. Since $\rho \in \mathcal{F}_{AF_{k+1}}$ it is defended against these attacks by arguments in $\mathcal{F}_{AF_k}$. These arguments are part of $GE_{AF}$ according to the induction step. Therefore, $\beta$ will also be in $GE_{AF}$.

$\rho \in GE_{AF_1}$, therefore for all $\alpha$ in $\mathcal{A}$ defeating $\rho$, there is some $\beta \in GE_{AF_1}$ such that $\beta$ defeats $\alpha$. $\beta$ will also be in $GE_{AF}$. Therefore, for any argument $\alpha \in \mathcal{A}$ defeating $\rho$, there exists $\beta \in GE_{AF}$ defeating $\alpha$.

Accordingly, we can show that for any $\beta \in \mathcal{B}$ defeating $\rho$, there exists an argument $\alpha \in GE_{AF}$ defeating $\beta$. Therefore, $\beta$ is defended against all defeats from $\{\rho\} \cup \mathcal{A} \cup \mathcal{B}$ by arguments in $GE_{AF}$. So $\rho \in GE_{AF}$. □

The following proposition asserts that for agents following confident strategies, with initially conflict-free argument sets, if a proposal is accepted by the dialogue, then it is acceptable with respect to the union of the agents' arguments.

**Proposition 19.** *If an iterated dispute between two agents i, j following confident strategies, terminates accepting a proposal argument* $\rho$*, then* $\rho$ *is in the grounded extension of the argumentation framework* $AF = \langle \mathcal{A} \cup \mathcal{B} \cup \{\rho\}, Defs \rangle$*, where* $\mathcal{A}$ *and* $\mathcal{B}$ *are the initial, finite and conflict-free argument sets for agent i and j respectively.*

*Proof.* Let *i* be the agent that made the accepted proposal. Consider the following argumentation frameworks: $AF^{PRO} = \langle \mathcal{A} \cup \mathcal{B}' \cup \{\rho\}, Defs \rangle$ and $AF^{OPP} = \langle \mathcal{B} \cup \mathcal{A}' \cup \{\rho\}, Defs \rangle$. $\mathcal{A} \cup \mathcal{B}'$ denotes the arguments *PRO* knew before initiating the final dispute and $\mathcal{B} \cup \mathcal{A}'$ is the set of all the arguments *OPP* knows after the dispute has terminated. $\mathcal{A}' \subseteq \mathcal{A}$ and $\mathcal{B}' \subseteq \mathcal{B}$. The agents follow confident strategies, so the proponent will propose arguments that are in the grounded extension of $AF^{PRO}$. According to Lemma 1 the proposal argument will be in the grounded extension of the opponent $GE_{AF^{OPP}}$ if a dispute terminates as a failing attack. According to Lemma 2 if the proposed argument is in $GE_{AF^{PRO}} \cap GE_{AF^{OPP}}$ then it is part of $GE_{AF}$. □

In the general case the proposal acceptance of iterated disputes is sound with respect to the agents' individual arguments. In the special case in which both agents

initial argument sets are individually consistent, accepted proposals are sound with respect to the union of the agents' arguments. This follows from the persuasion dialogue nature of the protocol, and shows that the focus of this protocol is to reach a mutually acceptable agreement. This result is also based on the exhaustive nature of the confident strategy, which ensures that agents investigate all relevant lines of defence and attack.

## 5.3 Arguing with Defeasible Basic Action Theories

Given that we now have a working protocol for iterated dispute dialogues, we focus on the internal structure of arguments. The planning knowledge of each agent is represented in defeasible situation calculus in the form of a defeasible basic action theory.

### 5.3.1 Plan Proposal Arguments

Plan proposal arguments are potential solutions to the multi-perspective cooperative planning problem. The claim of such arguments must convey that the situation term $S = do([A_1, A_2, \ldots, A_n], S_0)$ which corresponds to the proposed plan $\langle A_1, A_2, \ldots, A_n \rangle$, is an executable situation which satisfies the goal literals. This is denoted by the expression:

$$executable(S), goal(S).$$

As explained in Section 4.2, the expressions $executable(S)$ and $goal(S)$ are abbreviations. They are equivalent to:

- $Poss(A_1, S_0), \ldots, Poss(A_n, do([A_1, A_2, \ldots, A_{n-1}], S_0))$ and
- $G_1(S), \ldots, G_m(S)$ respectively.

The literal predicates $G_1, \ldots, G_m$ represent the shared goal.

Following the specification of arguments in defeasible logic programming, the claim of an argument is a ground literal predicate. Accordingly, in order to encode plan proposal arguments, we introduce the following additional axiom to every agent's DBAT:

$$Plan(S) \prec executable(S), goal(S),$$

The special literal $Plan(S)$ is added to the defeasible situation calculus language to represent that a situation term $S$ corresponds to a plan which achieves the goals of the agents and whose actions are applicable in sequence.

### 5.3.2 Dialogue Setting

The dialogue setting consists of two agents, each one holding a separate DBAT representing their planning knowledge. Given a DBAT $\mathcal{D}$, the set of corresponding arguments is bound by the relevant situation terms used for grounding $\mathcal{D}$. In order to restrict the setting to finite argument sets, we consider a reasonably large set of ground situation terms. Initially the set of available arguments for agent $i \in \{1,2\}$ is denoted by the set $Args_0^i$. $Args_0^i$ denotes the set of all arguments that can be constructed from theory $\mathcal{D}$ grounded for every situation term of reasonable length ε.

In practice, agents do not have to construct their initial argument sets as argument generation can be performed on demand. Also, for the purpose of a single dispute, agents need to consider only the arguments that are relevant to the ground situation term $S$, which appears in the claim of the plan proposal that initiated the dispute. Every argument which does not refer to a situation that is equal to or a predecessor to $S$ is irrelevant.

After the $k$th dispute, $i$ holds its initial argument set and the arguments introduced by the other agent during these disputes. This set is denoted by $Args_k^i$. The proposal move asserts that the proponent proposes arguments that are acceptable with respect to their argumentation framework. Also, if a proposal is found acceptable in the $k+1$th dispute of the dialogue, then this proposal is defended against every defeat presented by the opponent. As a result, the proposal argument is part of the grounded extension of the argumentation framework $AF_{PRO}^k = \langle Args_{PRO}^k, Defs \rangle$ and the framework $AF_{OPP}^k = \langle Args_{OPP}^k \cup \mathcal{A}, Defs \rangle$, where $\mathcal{A}$ are the arguments that the proponent presented during dispute $k+1$.

### 5.3.3 Dialogue with Conflict-Free Argument Sets

In the general case, since individual DBATs may be contradictory, the argument sets held by the agents are not conflict-free. However, if we restrict their initial argument sets we can extend the results of Proposition 19 to dialogues about plans.

#### 5.3.3.1 Arguing with Individually Acceptable Arguments

One way to ensure that the initial argument sets are conflict-free is to restrict them to the arguments that are acceptable with respect to individual theories. In this setting,

the argument set for agent $i$ is specified as follows:

$$\widehat{Args}_0^i = GE_{\langle Args_0^i, Defs\rangle}\,.$$

Following Proposition 19, every proposal ρ accepted by the dialogue is part of the grounded extension of the argumentation framework $\widehat{AF} = \langle \widehat{Args}, Defs\rangle$, where $\widehat{Args} = GE_{\langle Args_0^1, Defs\rangle} \cup GE_{\langle Args_0^2, Defs\rangle}$.

#### 5.3.3.2 Encoding Domains without Default Negation

The use of default negation in the frame part of successor state axioms is responsible for multiple contradictory derivations, even in theories which encode non-contradictory planning knowledge. Defeasible derivations treat default negated literals as assumptions and shift the burden of the evaluation of these assumptions to the argumentation process.

We can axiomatise the planning domain without the use of default negation, simply by replacing it with normal negation. Consider the following ground frame axiom for the literal $L(do(a,s))$, where $F_1$ and $F_2$ are fluent predicates:

$$L(do(A,S)) \multimap L(S), not\, F_1(S), not \sim F_2(S)\,.$$

The equivalent ground axiom with normal negation in the place of the default negation has the following form:

$$L(do(A,S)) \multimap L(S), \sim F_1(S), F_2(S)\,.$$

During the grounding process, and after the transformation of axioms into a well-formed structure, we simplify consecutive occurrences of normal negation symbols by removing them in pairs for as long as two negation symbols exist in sequence. Let a DBAT $\mathcal{D}$, we represent the corresponding theory obtained by replacing default negation as $\widetilde{\mathcal{D}}$.

In the general case, there are derivations made from $\mathcal{D}$ based on assumptions which cannot be derived from $\widetilde{\mathcal{D}}$. These derivations may correspond to undefeated arguments. For instance, let a ground literal $L(S)$ such that $\mathcal{D} \,|\!\sim L(S)$ and $\mathcal{D} \,|\!\sim \overline{L}(S)$. The use of normal negation in place of default negation strictly restricts the defeasible derivations that can be made from the theory. As a result, in this case it holds that $\widetilde{\mathcal{D}} \,|\!\sim L(S)$ and $\widetilde{\mathcal{D}} \,|\!\sim \overline{L}(S)$. In addition, assume that there exists an undefeated argument claiming $L'(do(A,S))$, which is constructed from $\mathcal{D}$ based on the assumption that

$not\,L(S)$. No corresponding argument can constructed from $\widetilde{\mathcal{D}}$ since there exists no derivation for $\overline{L}(S)$. Assuming that there are no other arguments claiming $L'(do(A,S))$ that can be constructed from $\mathcal{D}$ or $\widetilde{\mathcal{D}}$, we reach the conclusion that $\mathcal{D}\,|\!\approx L'(do(A,S))$ and $\widetilde{\mathcal{D}}\,|\!\not\approx L'(do(A,S))$.

It is clear from the previous example that, in the general case, the warrant results obtained from theories $\mathcal{D}$ and $\widetilde{\mathcal{D}}$ do not coincide. However, the translation is still useful in the special case in which theories are complete with respect to initial situation beliefs and contain one successor state axiom for every literal, with non-contradictory initial states and action effects. In other words, theories in which for every literal $L$ and every ground situation $S$ it holds that $\mathcal{D}\,|\!\approx L(S)$ or $\mathcal{D}\,|\!\approx \overline{L}(S)$. As a result, every argument produced from the default negation-free theory based on an assumption $not\,L(S)$ which cannot be derived (i.e. $\mathcal{D}\,|\!\not\approx L(S)$), is defeated by an acceptable argument claiming its complement $\overline{L}(S)$.

Standard planning theories fall into the above category. Given the corresponding DBAT without default negation $\widetilde{\mathcal{D}}$, for every ground literal $L(S)$ it holds that there exists a defeasible derivation of exclusively one of $L(S)$ and $\overline{L}(S)$. As a result, the sets of arguments constructed from $\widetilde{\mathcal{D}}$ is conflict-free. Therefore, if both agents hold standard planning theories, and a proposal argument is accepted by a dispute, then this argument $\rho$ is acceptable with respect to the argumentation framework consisting of the union of the agents' arguments. Formally in this case, for the plan proposal argument $\rho$, it holds that $\rho \in GE_{AF}$, where $AF = \langle Args, Defs \rangle$.

### 5.3.4 Belief Alignment

The iterated disputes dialogue protocol aligns the beliefs of the agents by allowing them to incorporate the arguments that were presented by the other agent to their own sets. This is particularly useful with respect to arguments that remained undefeated, or are used in the dialogue to successfully defeat the agent's proposal. By considering these arguments, the agents assert that future proposals will not be rejected for the same reasons that caused previous proposals to fail.

The belief alignment mechanism of the protocol works on the argument level. In order to take this process to the belief level, the agents must consider the supporting beliefs of the arguments. This leads to the generation of additional arguments based on each agent's individual knowledge combined with the new information that arises during the dialogue. Agents may incorporate new information to their belief sets after

the end of unsuccessful disputes. Alternatively, agents may re-calculate argument sets after every move made by the other party. The latter speeds-up the belief alignment process, but increases the required argument generation steps.

Arguments are supported by ground axioms which are only relevant to the specific proposal under discussion, and other proposals based on the same action subsequences. These ground rules have been constructed based on unground defeasible situation calculus axioms. In order to further facilitate the belief alignment process, agents can communicate, in conjunction to such ground beliefs, the unground axioms used to obtain these rules, as well as the associated preference values.

### 5.3.5 Minimal Plan Proposals

The support of a plan proposal argument is the minimal set of domain beliefs from which the claim of the proposal argument can be deduced. Depending on the length of the plan and the form of the axioms, the size of the support set can be extensive. In the worst case it can be comparable in size to the entire domain knowledge. In this section we present an alternative form of plan proposal arguments and discuss the advantages and drawbacks of such an approach.

The argument $\rho$ is a *minimal plan proposal argument* if $Claim(\rho) = Plan(S)$ and $Support(\rho) = \{Plan(S)\}$. If $\mathcal{D}_l^{PRO}$ are the beliefs for agent $i$ after iteration $l$, then in order for agent $i$ to present this argument, the following statement must hold: for all literals $X$ appearing in the abbreviations $executable(S)$ and $goal(S)$ it holds that $\mathcal{D}_l^{PRO} |\!\approx X$. Minimal plan proposal arguments present the belief that the plan holds without providing the support for this claim.

We extend our protocol, enabling the opponent to challenge the support of minimal proposal arguments, and the proponent to expand them accordingly. This is useful when the opponent has no reason to neither accept or object to the proposal and requires more information to evaluate it properly.

The *challenge* move challenges a future situation statement in the support of a plan proposal argument.

| $m_{l,k} = \langle challenge, OPP, X \rangle$ | |
|---|---|
| Conditions: | Effects: |
| $PreviousMoveType(\mathfrak{dl}) \in$ $\{proposal, expand, counter\}$ | $\mathcal{T}_{l,k} := T_{l,k-1}$ |
| $\mathcal{T}_k = \langle \rho \rangle$ | $\nu_{l,k} := null$ |
| $\exists L(S) \in Support(\rho)$ such that | $CS^{PRO}_{l,k} := CS^{PRO}_{l,k-1}$ |
| $\mathcal{D}^{OPP}_l \not\approx L(S)$ | $P_{l,k} := P_{l,k-1}$ |
| | $Q_{l,k} := Q_{l,k-1}$ |
| | $Args_{l,k} := Args_{l,k-1}$ |

The *expand* move can be used after a challenge move, extending the proposal argument and justifying the challenged support. This move works as one-step derivation.

| $m_{l,k} = \langle expand, PRO, \Phi \rangle$ | |
|---|---|
| Conditions: | Effects: |
| $\mu_k = \langle challenge, OPP, L(S) \rangle$ | $\mathcal{T}_{l,k} := \mathcal{T}_{l,k-1}$ with $\rho$ |
| $\Phi \cup Support(\rho)$ is non-contradictory | replaced by $\rho'$ |
| $\Phi \cup Support(\rho) \mid\sim L(S)$ | $V_{l,k} := \rho'$ |
| $\Phi \cup Support(\rho)$ is minimal | $CS^{PRO}_{l,k} := (CS^{PRO}_{l,k-1} \setminus \{\rho\}) \cup \{\rho'\}$ |
| | $P_{l,k} := (P_{l,k-1} \setminus \{\rho\}) \cup \{\rho'\}$ |
| | $Q_{l,k} := \emptyset$ |
| | $Args_{l,k} := \emptyset$ |
| | Where: |
| | $Claim(\rho') := Claim(\rho)$ |
| | $Support(\rho') :=$ $(Support(\rho) \setminus \{L(S)\}) \cup \Phi$ |

The confident strategy is extended accordingly:

$$\langle counter, backup, retract, challenge, expand, close, accept, propose, no\text{-}proposal, terminate \rangle .$$

A line of consecutive challenges and expansions can be continued until the support of the proposal includes statements about the initial state. Each expansion replaces a sentence with a set of axioms and sentences of the previous situation that are sufficient to derive it. The overall number of consecutive challenges and expansions is bound by the number of actions in the plan. The opponent can terminate a line of challenges and

expansions using a counter move. If the proponent does not defend the plan against all defeats, then the dispute terminates, and there is no need for the proponent to communicate the remaining support of $\rho$.

This extension is useful for proposals with extensive support. The benefit is that the proponent will need to expand the support of $\rho$ only for the statements for which the agents disagree. An extreme case in which this extension minimises the required communication is when the opponent immediately agrees with the claim of the plan proposal argument, without making any challenges or counter moves.

The modified protocol does not always produce sound results with respect to the union of both agents' arguments. The opponent may accept a minimal argument if the same conclusions are made from $\mathcal{D}_{OPP}$. However, the proponent's argument may be based on a literal $L(S)$ which is not warranted from the opponent's theory, and against which *OPP* holds a defeater, that cannot be in turn defeated. Equivalently, the proponent may hold arguments that can defeat the opponent's view on why the plan is acceptable. Therefore, the minimal proposal protocol may lead to incorrect conclusions if the agents have different reasons for accepting a proposal, and all these reasons are flawed. The following simple propositional example illustrates this issue:

**Example 5.** *Let agent i and agent j's theories* $\mathcal{B}^i = \{b, a \multimap b, \sim c\}$ *and* $\mathcal{B}^j = \{\sim b, a \multimap c, c\}$ *respectively. Also let the preference levels* $pref(\langle\{\sim c\}, \sim c\rangle) > pref(\langle\{c\}, c\rangle)$ *and* $pref(\langle\{\sim b\}, \sim b\rangle) > pref(\langle\{b\}, b\rangle)$. *We consider the preference value of the argument to be the lowest preference value of any one of the beliefs in its support. Both agents cannot counter and will not challenge the minimal proposal argument* $\langle\{a\}, a\rangle$, *since a is warranted from both agents' beliefs. However, in both cases the non-minimal corresponding proposal arguments* $\langle\{b, a \multimap b\}, a\rangle$ *and* $\langle\{c, a \multimap c\}, c\rangle$, *are not acceptable with respect to the union of the agents' argument sets.*

The modified version can substantially reduce the size of the proposal arguments, when agents disagree about a small subset of their knowledge. This protocol produces sound results with respect to both argumentation frameworks $\langle Args_{l-1}^{PRO} \cup \{\rho\}, Defs\rangle$, and $\langle Args_{l}^{OPP} \cup \{\rho\}, Defs\rangle$, for a minimal proposal argument $\rho$.

## 5.4 Multi-Party Iterated Disputes

When faced against problems that require the cooperation of multiple agents, the iterated disputes protocol requires the agents to argue in pairs. Such an approach is not always practical, since it requires multiple dialogues in order to reach agreement. In addition, the outcome of the overall process depends on the order these dialogues are conducted. During every dialogue, agents learn new information, which results in the construction of additional arguments. As a result, certain chains of arguments may never come up, since there may exist individual arguments which never become common knowledge.

### 5.4.1 The Multi-Party Iterated Disputes Protocol

In order to tackle the issues related to the two-party nature of the protocol, we present modifications that enable multiple agents to participate in one dialogue. The resulting mechanism allows every agent to present all relevant arguments, regardless if these arguments support or object to the evaluated proposal. The main idea behind this extension is that the roles of the proponent and the opponent are not assigned to a single agent for an entire dispute. On the contrary, a role is assumed temporarily, for just one move, by the agent who is better capable to progress the state of the dialogue.

More specifically, at the beginning of every turn, all agents advertise the legal move that they would play if they assumed the role which is active in the current turn. The player advertising the most prominent move assumes the role and progresses the state of the dispute by preforming this move. The selection of the move with the higher preference is based on the confident strategy according to the following principles:

1. Ordering over move types:

$$\langle counter, backup, retract, close, accept, propose, no\text{-}proposal, terminate \rangle .$$

2. For *backup* moves, the move with the highest depth $e$.

3. The preference level of the argument presented by the move, for moves of the same type.

4. If the above rules do not apply, pick an agent randomly.

The first two rules assert that the argumentation tree that corresponds to the dialogue is expanded in a depth-first manner for every potential plan proposal. This is

performed by the assignment of the highest preference to moves countering the last argument in the current path. If no agent can construct a legal *counter* move, then depending on whether the current turn is played by the proponent or the opponent, the *backup* and *retract* moves are prioritised. Backup moves enable the opponent to provide an alternative attack to an argument supporting the plan proposal. We prioritise the potential moves based on the depth of the node in which the attack is made. Defeats closer to the leaf of current line are preferred.

Multiple moves of the same type may be advertised (with equal $e$ in the case of *backup* moves). If these moves introduce an argument (i.e. they are *counter*, *backup*, *retract*, or *propose* moves), the mechanism selects the move presenting the argument with the highest preference order. Finally, if all moves are equivalent, one move is selected non-deterministically.

When multiple proposal moves are advertised, the move that proposes the plan with the highest preference is played. As a result, the algorithm is not sequential with respect to turn-taking. Agents that cannot contribute to the dispute are not assigned the role of the proponent and the proponent, and as a result their moves are not performed. However, due to the exhaustive nature of the dialogue, every opinion that is relevant to discovering and evaluating proposal arguments is always presented.

### 5.4.2 Properties

Similar to the two-party protocol, this process performs exhaustive search to the potential plan proposal arguments and the arguments supporting and defeating them. However, in the multi-party version, the virtual proponent and the opponent agents are equipped with an argument pool containing the union of the agents' argument sets. Given finite initial argument sets, multi-party dialogues always terminate.

**Proposition 20.** *Multi-party iterated disputes between between a group of agents* $N = \{1,2,\ldots,n\}$ *with finite initial sets of arguments and potential proposals always terminate.*

*Proof.* This follows from the termination of iterated disputes and the hypothesis that initial argument sets are finite. The rules prohibiting the proponent agent to repeat the same arguments in the same dispute line are imposed on the virtual proponent agent. The arguments that are available to the virtual proponent agent are bound by the union of the agents' arguments. Since this set is finite, disputes always terminate. Finally, since according to the hypothesis potential proposals are finite, and in every

new dispute a new proposal must be made, the number of iterated disputes is bound by the number of potential proposals. As a result, multi-party iterated disputes always terminate. □

The results of the dialogue are sound with respect to the set of arguments that can be generated from all agents. More specifically, an argument is accepted by a multi-party iterated dispute if and only if it is part of the grounded extension of the argumentation framework $\langle Args, Defs \rangle$, where $Args$ is the union of the arguments that can be generated from the union of the agents' individual theories.

**Proposition 21.** *Let a terminated multi-party iterated dispute between a group of agents $N = \{1, 2, \ldots, n\}$ with finite initial sets of arguments $Args_0^i$ for agent $i$. The final dispute is a failing attack against a proposal argument $\rho$ if and only if $\rho \in GE_{\langle Args \cup \{\rho\}, Defs \rangle}$, where $Args = \bigcup_{i \in N} Args_0^i$.*

*Proof.* We focus on the final dispute accepting $\rho$. Due to the specification of the confident strategy in multi-party iterated disputes, this dispute is equivalent with a two-party dispute between agents $A$ and $B$, such that $Args_A = Args_B = Args$, which both follow the standard confident strategy. In both cases, a counter move is prioritised if there exists an argument in $Args$ that defeats the last argument introduced by the other party. Also, in both cases, the move presenting the argument with the highest preference is selected. If a counter move is not available, and it is the opponent's turn to make a move, then $OPP$ searches to backtrack to the most recent argument in the current path of the dialogue that can be attacked. The same process is performed by the virtual opponent in the multi-party dialogue. If it is the turn of the proponent the retract move is played in both cases accordingly.

According to Lemma 1, if the dispute terminates as a failing attack on an argument $\rho$, then $\rho \in GE_{\langle Args_l^{OPP} \cup \{\rho\}, Defs \rangle}$, where $Args_l^{OPP}$ are the arguments that the opponent agent knows after the termination of the dispute. Since the agents initially hold equivalent argument sets, $Args_l^{OPP} = Args$. As a result, $\rho \in GE_{\langle Args \cup \{\rho\}, Defs \rangle}$. □

Without further modifications to the dialogue moves, this process is sound but incomplete. The specification of proposal moves requires every proposal to be acceptable with respect to the arguments of the agent presenting it. Consider the case in which a proposal argument is held only by one agent, and let this argument be unacceptable with respect to this agent's argument set. This proposal cannot be presented. If the agents collectively hold arguments that are acceptable and can support the proposal

argument against every attack, then this proposal argument is part of the grounded extension of the collective argumentation framework.

To ensure completeness, we introduce another proposal move (preferably with lower preference) enabling agents to present proposals that are not in the grounded extension of their individual argumentation sets. The result of the devised protocol is sound, since the agent making the proposal can in turn attack it. At the same time, the resulting protocol is complete, since agents may make proposals that are not acceptable with respect to their individual argument sets, but are acceptable with respect to the union of the agents' arguments.

Multi-party iterated disputes do not follow the turn-taking mechanism of iterated disputes, since they do not ensure that turn taking is alternated among the agents. Agents who hold the most relevant arguments to the state of the dispute are given priority. The selection of the most suitable party for the current role does not require additional centralised control. Information about the advertised moves may be broadcast from every agent to all other agents participating in the dialogue. The evaluation of move preference does not require information that is internal to the agents, and can be conducted based on the advertised moves alone.

The advantage of this protocol is twofold. On the one hand, it provides a mechanism for multi-party distributed argument evaluation. On the other, by allowing agents to present arguments both in favour and against a proposal, this method is complete. As a result, if an acceptable argument exists, then it is always identified. However, this process requires the agents to exchange all relevant arguments, including arguments that they believe to be unacceptable. As a result, it is more expensive than a standard iterated dispute, since it requires the exploration of every argumentation path, regardless of the agents' individual views.

In the two party setting with inconsistent individual argument sets, the multi-party version of the protocol is preferable to the standard version of iterated disputes. Due to the lack of explicit-turn taking with fixed roles, it allows agents to introduce arguments that both defeat and defend a proposal. This is important if argument paths exist that can be constructed from the union of the agents' argument sets which may never appear using the standard iterated disputes protocol.

## 5.5 Inquiry-Based Iterated Disputes

The results discussed so far are based on a notion of soundness that focuses on the argument level. More specifically, in order to evaluate the correctness of our protocols we investigated the outcomes of iterated disputes with respect to the arguments that can be constructed from individual agent theories.

A centralised argumentation mechanism operating over the union of the agents' beliefs $\mathcal{D}$ evaluates the correctness of the proposals with respect to the arguments that can be constructed from $\mathcal{D}$. This set is greater than the union of the argument set that can be generated from individual theories, since the generation of certain arguments from $\mathcal{D}$ may be based on beliefs that are distributed among the agents' theories. Such arguments are not available to any individual agent. In order to tackle this issue, we combine our protocol with the *argument inquiry* protocol presented by Black and Hunter (2007, 2009), and enable distributed argument generation.

### 5.5.1 Argument Inquiry Dialogues

The argument inquiry protocol is based on multiple nested dialogues that search the space of defeasible derivations while trying to construct arguments. The following moves are specified:

| **Move** | **Format** |
|---|---|
| *open* | $\langle open, x, r \rangle$ |
| *assert* | $\langle assert, x, \langle H, h \rangle \rangle$ |
| *close* | $\langle close, x, r \rangle$ |

The move *open* initiates a new inquiry dialogue which searches for a derivation of a defeasible rule (or a defeasible fact) $r$. The symbol $x$ denotes the agent making the move. For instance, the open move can initiate a new dialogue for the expression $a \multimap b, c$.

A *question store* is associated with each dialogue maintaining the literals that require support. Following the previous example, the move $\langle open, x, a \multimap b, c \rangle$ initiates a dialogue with the literals $b$ and $c$ in its question store, whereas the move $\langle open, x, c \rangle$ begins a dialogue with $c$ in its question store.

The agents use the assert move to present the arguments they can construct and

which claim a literal that is contained in the question store of the current dialogue. These arguments can be later used as sub-arguments for the arguments constructed for the main claim in question. A commitment store is associated with each agent, maintaining the beliefs (in terms of defeasible rules and facts) supporting the arguments asserted by this agent. These commitment stores are used for argument generation, since each agent can utilise both its private knowledge and the beliefs in the other party's commitment store.

The *close* move denotes that the agent does not hold any other meaningful piece of information related to the literals in the question store of the current dialogue. The current dialogue terminates if both agents make consecutive close moves. This is called a *matched-close*.

The *argument inquiry strategy* selects the next move for agents participating in the inquiry protocol. Assert moves are prioritised over open moves, which are in turn preferred over close moves.

As a result, agents initially present the arguments they can construct for every open question in the question store of the current dialogue. After every relevant argument has been presented, they open new dialogues following any rules they hold whose head is a literal contained in the question store. Every such rule initiates a new dialogue, which may lead to the construction of additional arguments that cannot be synthesised individually. This process results in nesting inquiry dialogues. When there is no further assert or open move that can be performed, the agents close the current dialogue and return to the previous one.

The outcome of an inquiry dialogue is the set of arguments that can be constructed from the union of the agents' commitment stores. Black and Hunter (2007, 2009) show that argument inquiry dialogues terminate and their outcome is equivalent to the arguments that can be constructed from the union of the participating agent's theories.

The inquiry argument protocol is designed for two-party argument generation. However, it can be easily extended to accommodate multiple-party dialogue, since the issues associated with multi-party dialogue are not present in this case due to the collaborative and exhaustive nature of the process (Black and Hunter, 2009).

### 5.5.2 Argument Inquiry in Iterated Disputes

Similar to our methods, argument inquiry protocols are defined on top of defeasible logic programming. The main differences in the structure of the agents' beliefs in

Black and Hunter (2007, 2009) and grounded defeasible basic action theories is that DBATs consider all beliefs to be defeasible rules, and default negation is allowed in the bodies of these rules. In DBATs, defeasible facts are represented as presumptions (i.e. defeasible rules with an empty body).

In order to allow the use of default negation, a minor modification must be made to the argument inquiry protocol. Default negated literals must not be added to question stores, since they are treated as assumptions. This is a minor modification which simplifies the argument generating process, since no new dialogues have to be introduced for default negated literals. The definition of arguments should also be adapted to account for default negation. The termination, soundness and completeness results of the protocol are not affected by this minor modification, since they still correspond to the equivalent argument generation process from a single collective theory.

Another difference between the two frameworks is the preference ordering over beliefs and arguments in DBATs. To this end, the assert move must be associated with the preference value of the presented argument. If another agent holds a different preference ordering that leads to a different preference value for this argument, a different assert move must be made. As a result, the protocol is modified so that argument uniqueness is defined in terms of support, claim and preference, instead of just support and claim.

The commitment stores for the inquiry dialogue are not equivalent to the commitment stores used in iterated disputes. First of all, they store different pieces of information, since the latter store arguments, rather than beliefs. Most importantly, they serve a different role. The commitment stores in iterated disputes are used to create lines of attack (and defence) against the evaluated proposal argument, whereas commitment stores in inquiry dialogue are used to extend the agents' theories with the beliefs presented by the other party.

#### 5.5.2.1 The Inquiry-Based Iterated Disputes Protocol

We enable nested argument inquiry dialogues within iterated disputes using the following additional moves: *inquire_proposal* and *inquire_attack*. The first initiates an argument inquiry dialogue of the form $\langle open, x, Plan(S) \multimap executable(S), goal(S) \rangle$, where $S$ is the ground situation corresponding to plan $\pi$, and $x$ is the agent initiating the dialogue. It has the following structure:

| $m_{l,k} = \langle inquire\_proposal, i, S \rangle$ |
|---|
| Conditions: |
| $PreviousMoveType(\mathbb{d}) = close$ or $\mathbb{d} = \langle\rangle$ |
| $\nexists m \in \mathcal{D}$ such that $m = \langle inquire\_proposal, x, S \rangle$, for any agent $x$ |
| Effects: |
| $Args^x_l := Args^x_{l-1} \cup Outcome(\mathbb{d}^{inq}_S)$, for every agent $x$ |

$Outcome(\mathbb{d}^{inq}_S)$ denotes the outcome of the inquiry dialogue initiated by agent $x$, for the defeasible rule $Plan(S) \prec executable(S), goal(S)$. Proposal inquiry moves may be performed at the end of a dispute in order to allow agents to collaboratively generate proposal arguments.

In a similar fashion, the agents can use *inquire_attack* moves to search for potential attackers against the most recent argument move.

| $m_{l,k} = \langle inquire\_attack, i, r \rangle$ |
|---|
| Conditions: |
| $PreviousMoveType(\mathbb{d}) \in \{propose, counter, backup\}$ |
| $r \in attacker\_claims(\nu^l_{k-1})$ |
| $\nexists m \in \mathcal{D}$ such that $m = \langle inquire\_attack, x, r \rangle$, for any agent $x$ |
| Effects: |
| $Args^x_l := Args^x_{l-1} \cup Outcome(\mathbb{d}^{inq}_\phi)$, for every agent $x$ |

The inquiry attack move conducts an inquiry dialogue for a possible point of attack against $\nu^l_{k-1}$. The elements of the set $attacker\_claims(\nu^l_{k-1})$ denote the claims of every potential argument attacking $\nu^l_{k-1}$. These are the complements of the claims of sub-arguments of $\nu^l_{k-1}$, the claim of $\nu^l_{k-1}$, or default negated literals appearing in the bodies of defeasible rules in $Support(\nu^l_{k-1})$. More specifically, this set is specified as follows:

$$attacker\_claims(\alpha) = \{\overline{\psi} \mid \alpha' \text{ is } \alpha \text{ or } \alpha' \text{ is a sub-argument of } \alpha, \text{ and } Claim(\alpha') = \psi\} \cup \{\psi \mid r \in Support(\alpha) \text{ and } not\,\psi \text{ appears in the body of } r\}.$$

The conditions of the *inquire_attack* move assert that the agents do not conduct the same inquire dialogue twice, since the necessary arguments are already part of their argument sets.

The *inquire_attack* move allows the agents to identify potential defeaters of the most recent argument presented by the other party. In order to enable the agent to present these arguments, turn-taking must allow the same player to make the next move. To achieve this we introduce a the *noop* move that a player is obliged to perform exactly after an *inquire_attack* move.

| $m_{l,k} = \langle noop, i \rangle$ |
|---|
| Conditions: |
| $PreviousMoveType(\mathfrak{d}) = inquire\_attack$ |

Argument inquiry dialogues are treated by the iterated disputes moves as black boxes. They achieve the alignment of the agents' argument sets regarding all arguments claiming a literal $r$ (which opened the inquiry dialogue) that can be collaboratively generated. Inquiry dialogues can be optimised by prohibiting the agents from opening argument inquiry sub-dialogues that have been opened again by a previous inquiry dialogue.

In order to assert that all relevant arguments are shared knowledge, the agents must initiate an inquiry dialogue for every potential proposal and point of attack of any argument presented during an iterated dispute. This leads to an exhaustive approach which ensures that the results of the dialogue are sound and complete with respect to the arguments that can be generated from the agents' collective beliefs. To achieve this, we adapt the specification of the confident strategy priority ordering as follows:

$$\langle noop, inquire\_attack, counter, backup, retract, close, accept, \\ propose, inquire\_proposal, no\text{-}proposal, terminate \rangle .$$

The *inquire_attack* move is preferred over every argument move in the protocol. This asserts that before presenting a defeater, agents have every defeater that can be generated from the union the agents' theories at their disposal.

#### 5.5.2.2 Properties

The exhaustive use of the *inquire_attack* move ensures that accepted proposals are sound with respect to the arguments that can be generated from the union of the agents' beliefs.

**Proposition 22.** *Let a terminated inquiry-based iterated dispute $\mathcal{D}$ between two agents $i$ and $j$ following confident strategies. Also, assume that $\rho$ is the proposal of the final*

*dispute l of the iterated dispute. Dispute l terminates as a failing attack on* $\rho$ *if and only if* $\rho \in GE_{AF}$*, where* $AF = \langle Args, Defs \rangle$*, Args are the arguments that can be generated from the union of the agents beliefs and Defs denotes all defeat relationships between them.*

*Proof.* Assume that the final dispute $l$ is a failing attack. Prior to any counter moves made by the opponent, due to the specification of the confident strategy for inquiry-based iterated disputes, the opponent used multiple *inquire_attack* moves to find all potential defeaters against the most recent argument presented by the proponent. As a result, for every argument $A$ presented by the proponent in the current path, all defeaters of $A$ are known to the opponent in $l$. Let $Args_l^{OPP}$ be the arguments available to the opponent after $l$. $Args_l^{OPP} = Args_{l-1}^{OPP} \cup CS_l^{PRO} \cup inquiry\_outcomes(l)$, where $inquiry\_outcomes(l)$ are the outcomes of the inquiry dialogues conducted in $l$. There is no argument in $Args \setminus Args_l^{OPP}$ defeating an argument in the path form $\rho$ to $\nu_k^l$ presented by the proponent. As a result, the opponent using the exhaustive confident strategy has presented every argument in $Args_l^{OPP}$ that defeats the proponents' arguments. Therefore, the inquiry-based dispute emulates a standard dispute in which the arguments that are initially available to the opponent are $Args_{l-1}^{OPP} \cup inquiry\_outcomes(l)$. According to Lemma 1, $\rho \in GE_{\langle Args_{l-1}^{OPP} \cup inquiry\_outcomes(l), Defs \rangle}$. Since $\rho \in CS_l^{PRO}$ and $CS_l^{PRO}$ is conflict-free, $\rho \in GE_{\langle Args_l^{OPP}, Defs \rangle}$. Finally, since no argument in $Args \setminus Args_l^{OPP}$ defeats $\rho$ or the arguments that defend $\rho$, it holds that $\rho \in GE_{\langle Args, Defs \rangle}$.

If $l$ is a failing defence of $\rho$, there exists a tree in which the defeaters presented by the opponent cannot be countered by arguments presented by the proponent. For every argument presented by opponent, the proponent knows every counter-argument from $Args$. Since the exhaustive strategy enables the proponent to present any relevant argument, there exists no set of arguments defending the proposal against the arguments presenting by the opponent. As a result $\rho \notin GE_{AF}$. □

Termination of the process requires bounding exchanged proposals to a finite set.

**Proposition 23.** *Inquiry-based iterated disputes between two agents i and j who follow the confident strategy and are bound to proposing plans of maximum length* $\varepsilon$ *always terminate.*

*Proof.* The proof follows from Proposition 18 and from bounding the potential proposal and proposal inquiry moves to finite situation terms. Termination of inquiry dialogues is based on the results of Black and Hunter (2007, 2009), the finite size of

the agents' theories and the fact that the protocol of inquiry-based iterated disputes prohibits the agents to make repeated inquiry moves for the same claim. □

Completeness with respect to a finite set of plans is guaranteed by exhaustive use of the proposal inquiry move. However, to achieve this, the proposal move has to be modified, similar to the previous section, to enable agents present proposals that are not acceptable with respect to their individual arguments.

**Corollary 1.** *Let $\mathcal{D}$ an inquiry-based iterated dispute between two agents $i$ and $j$ following confident strategies bound to plans of maximum predefined length. If the iterated dispute fails then there exists no proposal argument P that can be constructed from the union of the agents' beliefs such that $\rho \in GE_{AF}$, where $AF = \langle Args, Defs \rangle$ is the argumentation framework containing all arguments that can be constructed from the union of the agents' beliefs.*

Proof follows from the use of the exhaustive *inquire_proposal* move which asserts that the agents share knowledge regarding every argument regarding a proposal that can be constructed from their collective beliefs, and the completeness results of argument inquiry dialogues (Black and Hunter, 2007, 2009).

The distributed argument generation of every potential plan (bound to a threshold on the length of the plan) is a very expensive process. As a rule of thumb, the agents can prioritise proposal inquires regarding plans whose success can be derived from their individual beliefs, then plans whose failure does not derive from their individual beliefs, and then everything else. The first category are plans that have been proposed earlier in the dialogue process unsuccessfully. Using proposal inquiry the agents can collaboratively search for an alternative support which potentially formulates an acceptable argument. The second category searches for plans against which the agents have no objections, even though their success does not derive from any individual theory. More specifically, given an agent theory $\mathcal{D}_i$, such plans correspond to a situation $S$ such that:

- $\mathcal{D} \not\approx Plan(S) \prec executable(S), goal(S)$,
- $\forall S'$ that are predecessor to $S$, $\mathcal{D} \not\approx \sim Poss(A', S')$, where $A'$ is the action applied in $S'$ according to the plan, and
- $\forall G_j(S)$ appearing in the abbreviation $goal(S)$, $\mathcal{D} \not\approx \sim G_j(S)$.

Finally, if no plan is accepted, the agents may search the space of potential proposals exhaustively.

## 5.6 Arguing with Basic Action Theories

The abstract nature of iterated disputes allows its use in conjunction with different underlying logical formalisms. In order to illustrate this, we present an alternative instantiation of the protocol of iterated disputes based on standard Reiter-style situation calculus (Reiter, 2001).

Compared to the defeasible situation calculus variant of the iterated disputes dialogue protocol, this version offers a more expressive logical formalism for the specification of the planning domain. However, it assumes that individual theories are non-contradictory and agents hold one successor state axiom for every fluent predicate in their theories. Similar to DBATs, a preference ordering is associated with the contents of the agents' initial situation beliefs and domain axioms. The resolution of contradictions is based on the selection of the initial state belief or axiom with the highest preference ordering. As a result, the aggregation of agents' axioms, which is implicitly performed while reasoning with DBATs, is not supported.

### 5.6.1 Arguments

The arguments held by the agents contain statements in the language of situation calculus. We employ an intuitive argument definition from the literature (Amgoud and Cayrol, 1998; Amgoud et al., 2000a). Arguments are defined based on an inference procedure $\vdash$ in a knowledge base, and reinterpret the defeat relation using logical contradiction:

**Definition 32.** Arguments *for agent* $i \in \{1,2\}$ *are pairs* $\alpha = \langle H, h \rangle$*, where* $H \subseteq 2^{\mathcal{L}_{SitCal}}$*, where* $\mathcal{L}_{SitCal}$ *is the set of all well-formed situation calculus sentences.*

1. *H is consistent (i.e.* $H \nvdash \bot$*),*
2. $H \vdash h$*,*
3. *H is minimal (no subset of H satisfies both (1) and (2)).*

*H is called the support of the argument and h its conclusion. We will also use the following notation:* $Support(\alpha) = H$*, and* $Claim(\alpha) = h$*. The preference level of an argument* $\alpha$ *is denoted by* $pref(A)$*, and is the minimum preference level of a statement in* $Support(\alpha)$*.*

**Definition 33.** *An argument* $\alpha_1 = \langle H_1, h_1 \rangle$ defeats *an argument* $\alpha_2 = \langle H_2, h_2 \rangle$*, denoted* $\alpha_1 \rightarrow \alpha_2$*, if* $pref(A_1) \geq pref(A_2)$ *and, either there exists* $h'$ *in* $H_2 \cup \{h_2\}$ *such that* $h_1 \equiv \neg h'$ *or* $h_1 = (F \equiv \Psi)$, $\phi = (F \equiv \Phi)$ *and* $\Phi \neq \Psi$*, where* $F$ *is a predicate symbol.*

The defeat relation considers contradictory beliefs and formulas providing different definitions of the same symbol. Reasoning about actions is based on the axioms representing the domain. It is essential that the domain theory does not include different axioms regarding the same predicate.

### 5.6.2 Planning Knowledge

The *domain knowledge* for agent $i$ after dispute $k$, is a set of beliefs $\mathcal{B}_k^i$, generated using the arguments in the grounded extension of $AF_k^i$, $GE_{AF_k^i}$, as illustrated below:

**Algorithm 15:** Algorithm for the construction of the domain knowledge bases

Compute $GE_{AF_k^i}$, for $AF_k^i := \langle Args_k^i, Defs \rangle$;
$\mathcal{B}_k^i := \{h \mid \nu \in GE_{AF_k^i} \wedge Claim(\nu) = h\}$;
**forall the** $h \in \mathcal{B}_k^i$ **do**
    $pref(h) :=$ maximum preference of an argument $\nu$ with $Claim(\nu) = h$;
**return** $\mathcal{B}_k^i$;

We consider the initial argument sets of the agents to be conflict-free. If all of their claims are basic action theory statements, all future domain beliefs constructed by this algorithm will be basic action theories.

### 5.6.3 Plan Proposals

A plan for a goal $G$ is represented in situation calculus by the statement: $executable(S_\pi) \wedge G(S_\pi)$. $S_\pi$ is a variable-free situation term representing the history for the execution of the actions of the plan in sequence, and $executable(s_\pi) \stackrel{\text{def}}{=} (\forall a, s^*).do(a, s^*) \sqsubseteq s_\pi \supset Poss(a, s^*)$. A consequence of the definition of a plan and the foundational axioms for situations is that $\forall a, s \,.\, executable(do(a, s)) \equiv executable(s) \wedge Poss(a, s)$.

**Definition 34.** *Plan proposal arguments for agent* $i$ *after dispute* $k$ *for a shared goal* $G$*, are all arguments* $\rho$ *s.t. i)* $Claim(\rho) = G(S_\pi) \wedge executable(S_\pi)$*, with* $S_0 \sqsubset S_\pi$*, ii)* $Support(\rho)$ *is the minimal subset of* $\mathcal{B}_k^i$ *such that* $Support(\rho) \vdash Claim(\rho)$.

If $\rho$ is a plan proposal argument then $\rho \in \mathcal{P}$. The preference level of a plan proposal argument is equal to the lowest preference level of the claims in its support.

The agents can obtain the support set of a plan proposal argument using Reiter's *regression operator* $\mathcal{R}$. The regression operator eliminates statements with complex situation terms by replacing them with logically equivalent statements that refer to situations closer to the initial state. The process is repeated until all fluents in the statement refer to the initial situation. The logical equivalence follows from the relevant action preconditions and the successor state axioms. A detailed analysis of the regression operator can be found in Reiter (2001).

The support of a proposal argument contains the minimal subset of domain beliefs and unique names assumptions sufficient to infer $\mathcal{R}[claim(\alpha)]$, as well as the equivalencies that were employed by the regression operator.

**Example 6.** *Using the axiom* $\forall a,s \,.\, F(do(a,s)) \equiv (a = A_1) \vee (a = A_2) \vee (f(s) \wedge a \neq A_3)$, $\mathcal{R}[F(do(A_4,S_0))]$ *returns* $A_4 = A_1 \vee A_4 = A_2 \vee F(S0) \wedge A_4 \neq A_3$, *which can be simplified to* $F(S_0)$.

The following proposition asserts that for all plans that can be constructed from an agent's planning knowledge, the plan proposal arguments for these plans will be part of the grounded extension of $\langle Args_l^i \cup \{\rho\}, Defs\rangle$, where $Args_l^i$ are the arguments known to agent $i$ after dispute $d_l$.

**Proposition 24.** *If* $\rho$ *is a plan proposal argument with* $Claim(\rho) = G(S_\pi) \wedge executable(S_\pi)$, *constructed in iteration k by agent i, then* $\rho \in GE_{\langle Args_l^i \cup \{\rho\}, Defs\rangle}$.

*Proof.* The proposal argument $\rho$ does not defeat any argument in $GE_{\langle Args, Defs\rangle}$, since $Claim(\rho)$ refers to a future situation $S_\pi$ with $S_0 \sqsubset S_\pi$, whereas all statements in the claim or support of arguments in $GE_{\langle Args, Defs\rangle}$ are initial situation statements, non-fluent statements and domain axioms. Therefore, $\forall \alpha \in GE_{\langle Args_l^i, Defs\rangle}$, it will be the case that $\alpha \in GE_{\langle Args_l^i \cup \{\rho\}, Defs\rangle}$. All statements in the support of $\rho$ are claims of arguments in $GE_{\langle Args_l^i \cup \{\rho\}, Defs\rangle}$, therefore any defeats against $P$ will be defended by arguments in the grounded extension. So $\rho \in GE_{\langle Args_l^i \cup \{\rho\}, Defs\rangle}$. □

Proposition 25 extends the result of Proposition 19 to dialogues about plans.

**Proposition 25.** *If an iterated dispute terminates with both agents accepting a plan proposal* $\rho$, *then* $\rho \in GE_{\langle Args \cup P, Defs\rangle}$, *for agents with initially conflict-free argument sets following confident strategies.*

Proof follows from propositions 19 and 24. This proposition asserts that under the aforementioned assumptions, if a plan is proposed by the proponent and accepted by the opponent though the dialogue, then this plan is acceptable with respect to the union of the arguments of all agents.

Plan generation is conducted as planning. Planning can be performed in situation calculus (Reiter, 2001), or by an equivalent PDDL representation (Röger et al., 2008) using a standard planner.

## 5.7 Summary

This chapter presents an in-depth analysis of a dialogue-based, distributed solution to the problem of multi-perspective collaborative planning. This analysis is based on the protocol of iterated disputes, which allows two agents to discuss potential proposals and reach conclusions about their acceptability. The protocol is based on abstract argumentation. Given a finite set of arguments it guarantees termination. In the general case, accepted arguments are sound from each agent's individual perspective.

The standard version of the protocol is based on persuasion. It assigns specific roles for every specific proposal to agents, and restricts the arguments that they can present according to their role. In the special case that the agents initially hold conflict-free argument sets, the protocol guarantees that successful proposals are acceptable with respect to the union of the agents' arguments.

In order to relax the assumptions about the two-party nature of the approach, we modified the protocol enabling multi-party discourse. The resulting version of the protocol enables the exchange of roles, allowing the agent who holds the most prominent move to make it at any point in the dispute. The results showed that accepted proposals of the multi-party version of the protocol are sound with respect to the union of the agents arguments.

In order to guarantee sound and complete results in the belief level, we presented a two-person inquiry-based version of the protocol. In this case, agents are allowed to enter inquiry dialogues, and collaboratively generate arguments for specific claims. This process is very expensive, but allows sound and complete (bound to plans of specific length) results with respect to the arguments that can be constructed form the union of the agents' beliefs.

Iterated disputes focus on agreement and allow agents to directly evaluate potential proposals which they individually consider to be viable. As a result, they do not col-

laborative explore the argument and belief space. Multi-party iterated disputes allow the agents to collaboratively explore the argument space, offering a stronger notion of soundness. Compared to the other versions of our protocol, multi-party iterated disputes do not guarantee equal opportunities to agents to present their arguments. Inquiry-based iterated disputes are the most expensive version of our protocol, since agents are required to exhaustively search the defeasible derivation space for potential arguments. However, inquiry-based iterated disputes offer an even stronger notion of soundness since these results are equivalent to the ones obtained from a centralised argumentation process operating on the union of the agents' beliefs.

In order to show the generic nature of iterated disputes, we presented concrete versions of the protocol for both defeasible basic action theories and standard basic action theories. Situation calculus offers higher expressive power then defeasible situation calculus. However, this version of the protocol requires individual theories to be non-contradictory, and does not allow agents to aggregate the conclusions made using different axioms.

The main contributions of this chapter are summarised as follows:

- A family of abstract protocols for dialogue-based collaborative agreement on a proposal.
- Formal analysis of termination, soundness and completeness of the presented protocols.
- Concretisation of the protocol for the problem of multi-perspective cooperative planning based on standard and defeasible basic action theories.

# Chapter 6

# Evaluation

Following our hypothesis, this chapter conducts a comprehensive evaluation of our methods, which investigates the following questions:

1. Is the problem of multi-perspective cooperative planning common?
2. Is argumentation theory suitable for the specification of the MPCP problem?
3. What is the quality of the proposed solution to the MPCP problem?

In order to answer these questions, we review our analytical results, empirical experimentation with benchmark planning domains, and discussion about MPCP problem instances in important real-world domains. The latter is not a formal evaluation of our methods, but a discussion regarding the applicability potential of our approach.

## 6.1 Review of Analytical Results

This section reviews the results of our analytical evaluation. First of all, we look into the proposed formalism, and investigate its ability to represent MPCP problems in a succinct manner. In order to evaluate the suitability of the formalism, we motivate our design choices, and compare the formalism with standard languages for reasoning about dynamic domains. The second part of this section focuses on the proposed reasoning mechanisms, and summarises their formal properties that were presented in the previous chapters.

### 6.1.1 Formalism

This section evaluates the expressiveness of defeasible situation calculus and defeasible basic action theories. We focus on the following questions:

- Can this formalism represent MPCP problems?
- Does it produce succinct representations?

In order to adequately answer these questions we look into the main features of the language to investigate whether the formalism fulfils its primary purpose. We overview important aspects of our formalism and, in order to highlight our design choices and illustrate its expressive power, compare it to formalisms from the relevant literature.

#### 6.1.1.1 Formalism Adequacy

The question whether the proposed formalism is sufficient to represent MPCP problems has two sides. First of all, it is related to whether the $\mathcal{L}_{defsitcal}$ language and DBATs are adequate to encode set-theoretic MPCP problems. In addition, since the main purpose behind the introduction of the formalism is to provide the means to decide which conclusions are rational to accept, we look into the limitations of the use of grounded argumentation semantics.

The formal results described in Chapter 3, and particularly Proposition 13, explain that reasoning about plans with basic action theories emulates state-based derivation, and specifies an argumentation-based reasoning mechanism on top of that. DBATs are strictly more expressive than MPCP problems. As a result, the expressiveness of DBATs is sufficient to represent MPCP problems.

$\mathcal{L}_{defsitcal}$ is based on a combination of situation calculus and defeasible logic programming. It has been designed to allow reasoning about dynamic domains with contradictory theories.

The resolution of contradictions depends on the employed argumentation semantics. The presented analysis of the specification of the problem, and our proposed solution mechanism, are based on grounded (sceptical) argumentation semantics. These semantics have been selected for their practicality (i.e. they are relatively inexpensive to calculate), and for their sceptical nature, which is essential in safety-critical domains.

We designed our methods in a modular way to enable the use of different argumentation semantics. The only method presented that is more tightly coupled to the

specific semantics is the iterated disputes protocol. However, even in this case modifying the protocol to follow different argumentation semantics is straightforward, since it is based on two-party immediate response disputes, and alterations to the protocol for different argumentation semantics have been proposed in the literature (Vreeswijk and Prakken, 2000).

Grounded semantics have limitations in situations in which reasoning by cases is required. In order to investigate the importance of reasoning by cases and expose any limitations of grounded argumentation semantics in relation to reasoning with DBATs, we investigate special cases.

**Example 7.** *Assume a specification of the action flip(switch), containing the conditional effects* $\langle\{Up(switch)\}, Lit(lamp)\rangle$ *and* $\langle\{\neg Up(switch)\}, Lit(lamp)\rangle$*. This specification corresponds to the following successor state axiom grounded for the objects* $switch = Switch$ *and* $lamp = Lamp$ *in situation* $S_1 = do(flip(Switch), S_0)$*:*

$$Lit(Lamp, S_1) \multimap Up(Switch, S_0); \sim Up(Switch, S_0); Lit(Lamp, S_0)$$
$$\sim Lit(Lamp, S_1) \multimap \sim Lit(Lamp, S_0), not\ Up(Switch, S_0), not \sim Up(Switch, S_0)$$

*Let the initial state be* $\{Up(Switch), \neg Up(Switch), \neg Lit(Lamp)\}$*, and assume that all beliefs are equally preferred. In this case the following arguments are generated:*

$\alpha_1 : \langle\{Up(Switch, S_0)\}, Up(Switch, S_0)\rangle$,

$\alpha_2 : \langle\{\sim Up(Switch, S_0)\}, \sim Up(Switch, S_0)\rangle$,

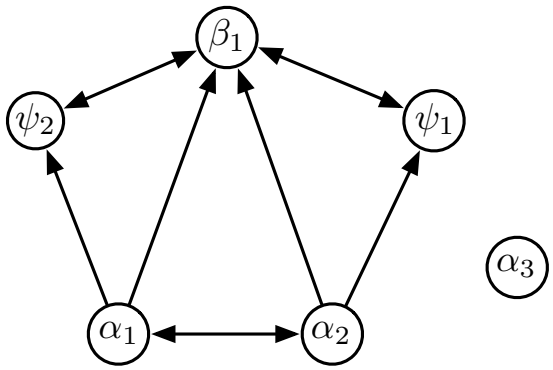
$\alpha_3 : \langle\{\sim Lit(Lamp, S_0)\}, \sim Lit(Lamp, S_0)\rangle$,

$\beta_1 : \langle\{\sim Lit(Lamp, S_1) \multimap \sim Lit(Lamp, S_0), not\ Up(Switch, S_0), not \sim Up(Switch, S_0),$
$\sim Lit(Lamp, S_0)\}, \sim Lit(Lamp, S_1)\rangle$,

$\psi_1 : \langle\{Lit(Lamp, S_1) \multimap Up(Switch, S_0),\ \ Up(Switch, S_0)\}, Lit(Lamp, S_1)\rangle$,

$\psi_2 : \langle\{Lit(Lamp, S_1) \multimap \sim Up(Switch, S_0),\ \ \sim Up(Switch, S_0)\}, \sim Lit(Lamp, S_1)\rangle$.

*The defeats among the arguments are depicted as follows:*




*The only argument that is not attacked is* $\alpha_3$*, and this argument does not attack any other argument. Therefore, following grounded acceptability semantics, the grounded extension contains only* $\alpha_3$*.*

*The argumentation framework contains two preferred extensions:* $\{\alpha_1, \psi_1, \alpha_3\}$ *and* $\{\alpha_2, \psi_2, \alpha_3\}$*. As a result, the only argument that is sceptically acceptable with respect to preferred argumentation semantics is still* $\alpha_3$*, even though there exist arguments in both extensions with the claim* $Lit(Lamp, S_1)$ *(i.e. arguments* $\psi_1$ *and* $\psi_2$*).*

In order to reach different conclusions in this case sceptical preferred argumentation semantics can be used combined with a modified version of warrant that accepts a literal if there exists an argument in every preferred extension that has this literal as its claim. This is a special case of a problematic situation in which a specification contains conditional effects that produce the same literal, which specify contradicting conditions. The following example illustrates a more elaborate case with contradictory operator specifications.

**Example 8.** *Consider a domain with a lamp and a switch. The initial situation beliefs held by the agents involve ambiguity regarding whether the switch is initially in the "on" position. In addition, there are conflicting beliefs regarding the specification of the effects of flipping the switch. More specifically, according to one specification flicking the switch turns the lamp on, whereas according to the other the lamp is switched on when the switch is pushed down. Assume the following initial situation beliefs:* $\sim Lit(Lamp, S_0)$*,* $Up(Switch, S_0)$ *and* $\sim Up(Switch, S_0)$*. Also, assume the*

*following ground effect axioms:*

$$Lit(Lamp, push(Switch, S_0)) \prec Up(Switch, S_0)$$
$$Lit(Lamp, push(Switch, S_0)) \prec \sim Up(Switch, S_0)$$
$$\sim Lit(Lamp, push(Switch, S_0)) \prec \sim Lit(Lamp, S_0), not \sim Up(Switch, S_0)$$
$$\sim Lit(Lamp, push(Switch, S_0)) \prec \sim Lit(Lamp, S_0), not\, Up(Switch, S_0)$$

*Accordingly, the following arguments are constructed:*

$\alpha_1 : \langle \{Up(Switch, S_0)\}, Up(Switch, S_0) \rangle,$

$\alpha_2 : \langle \{\sim Up(Switch, S_0)\}, \sim Up(Switch, S_0) \rangle,$

$\alpha_3 : \langle \{\sim Lit(Lamp, S_0)\}, \sim Lit(Lamp, S_0) \rangle,$

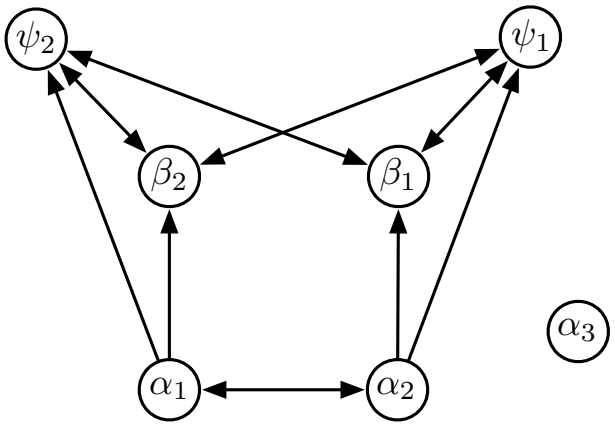
$\beta_1 : \langle \{Lit(Lamp, push(Switch, S_0)) \prec Up(Switch, S_0), Up(Switch, S_0)\},$
$Lit(Lamp, push(Switch, S_0)) \rangle,$

$\beta_2 : \langle \{Lit(Lamp, push(Switch, S_0)) \prec \sim Up(Switch, S_0), \sim Up(Switch, S_0)\},$
$Lit(Lamp, push(Switch, S_0)) \rangle,$

$\psi_1 : \langle \{\sim Lit(Lamp, push(Switch, S_0)) \prec \sim Lit(Lamp, S_0), not \sim Up(Switch, S_0)\},$
$\sim Lit(Lamp, push(Switch, S_0)) \rangle,$

$\psi_2 : \langle \{\sim Lit(Lamp, push(Switch, S_0)) \prec \sim Lit(Lamp, S_0), not\, Up(Switch, S_0)\},$
$\sim Lit(Lamp, push(Switch, S_0)) \rangle.$

*The attacks among the arguments are depicted as follows:*



*Following grounded argumentation semantics, the only argument that is not attacked is $\alpha_3$, which does not attack any other argument. Therefore, no other argument is part of the grounded extension.*

Even in such an elaborate example, reasoning by cases does not lead to better results for sceptical agents. The preferred extension of the above argumentation framework are $\{\alpha_1, \beta_1, \alpha_3\}$, $\{\alpha_1, \psi_2, \alpha_3\}$, $\{\alpha_2, \beta_2, \alpha_3\}$ and $\{\alpha_2, \psi_1, \alpha_3\}$. Therefore, the only argument that is sceptically acceptable with respect to preferred argumentation semantics is $\alpha_3$.

Grounded semantics impose a strict notion of acceptability, especially when agents hold views that contradict each other directly, and lack any meaningful deductive knowledge that can help in the resolution of these contradictions. In order to overcome such situations we allow the association of beliefs and arguments with preference orderings. This preference ordering is not essential to our methods, but is useful in cases in which no decision can be made based purely on the agents' beliefs.

The origin of the preference ordering and its semantics are domain specific. No restrictions are imposed by our framework on rules governing this ordering. It may be based on the credibility of the origin of the information, the quality of sensors responsible for observations, how outdated the information is, or may even be based on the aggregation of multiple factors.

An alternative way to calculate argument preference, if such meta-information is not present, is based on information about the axioms used to derive the conclusion of the argument. For instance, based on domain specific knowledge, agents may prefer arguments made using effect axioms to conclusions reached using frame rules. In this case, the decisions regarding arguments with contradicting conclusions that are not undercut are made based on the number of frame rules used in the derivation of the claim. Another option is to count the number of conditions stated in the axioms that are used to make a derivation. Depending on domain specific knowledge, agents may prefer more specific rules, which may entail a more accurate view of the planning domain.

#### 6.1.1.2 Expressiveness and Tractability Trade-off

The practicality of the approach is related to the expressiveness of the formalism and its ability to offer scalable synthesis of warranted plans. These notions are closely related. There is a fundamental trade-off between the expressiveness of the represen-

tation and computational tractability of the reasoning scheme. Expressive formalisms offer succinct representations, but often lead to intractable reasoning. On the other hand, propositional representations simplify the reasoning process but lead to impractical theory sizes.

This tension is apparent in MPCP. Our design choices focus on a tractable middle ground which is practical with respect to both representation and reasoning. We use variables and conditional effects, which allow succinct planning domain representations, but do not offer the complete expressive power of standard situation calculus. Reiter-style situation calculus is based on higher-order logic. The higher-order elements of the language are introduced by the fundamental axioms for situations. In practice, reasoning for ground queries can be performed in first-order theories using the regression operator to translate queries about successor situations into logically equivalent queries referring exclusively to the initial situation.

Reasoning with first-order theories is not tractable in the general case. Reasoning with contradictory first-order theories is an open area of research (Besnard and Hunter, 2005). In situation calculus theories, this would requires the use of the regression operator based on the assumption that successor state axioms are complete. We have shown how these can be used for dealing with multiple perspectives in Section 5.6. The limitation of this mechanism is that when possible alternative axioms exist, it reasons by selecting one axiom. This process cannot aggregate the results of multiple axioms. On the contrary, our defeasible situation calculus formalism allows the reasoning mechanism to consider the conclusions made using multiple axioms for the same fluent, thus implicitly aggregating the results that can be concluded by investigating all alternatives.

With respect to reasoning tractability, a naive, propositional argumentation-based approach to planning with DBATs is not feasible, since it leads to an impractical theory size. However, by utilising the inherent characteristics of the planning domain, we have shown that planning can be performed as a series of ground queries, and that these queries can be accurately evaluated from restricted ground theories. We investigate this further in Section 6.2, where we evaluate our methods experimentally in benchmark planning domains.

### 6.1.2 Comparison to Related Work

This section compares the proposed formalism with relevant approaches from the literature. We present this comparison to highlight the expressiveness of our formalism and explain the design choices behind our approach.

#### 6.1.2.1 Classical Planning

DBATs enable the succinct representation of planning domains by allowing two very important features of set-theoretic planning representations: variables and conditional effects. DBATs are strictly more expressive than MPCP problems, which adapt a representation based on STRIPS, extended to allow contradicting information, variables, negative conditions, and conditional effects. Extended DBATs can also encode ramifications and state constraints that can be represented as extended defeasible rules. Domains in PDDL that require additional expressive power need to be translated into a simplified theory (Nebel, 2000) before applying our methods. In their current form DBATs cannot represent functions which could be interesting in metric planning domains, or preferences which are relevant in a strategic setting.

#### 6.1.2.2 Situation Calculus

DBATs are syntactically equivalent to the standard Prolog implementation of Reiter-style situation calculus theories. However, there is one important distinction: DBATs are written in terms of literals, instead of atoms, to allow the specification of rules describing contradictory information.

Reiter (2001) offers a translation mechanism, based on a revised form of the Lloyd-Topor transformations, which allows the translation of basic action theories to equivalent theories that can be directly implemented in Prolog. Following these transformations, we presented a translation mechanism that deals with disjunctive expressions, and negation symbols preceding expressions (and not just literals), that cannot appear in well-formed extended defeasible rules. The presented mechanism may be extended to incorporate additional rules from Reiter (2001) that are capable of handling existential and universal qualifications in the bodies of axioms. In a similar fashion we can rely on standard planning translation mechanisms that can encode rich planning theories into standard STRIPS.

Another limitation of our approach is that it focuses on answering ground queries. As we have shown, these queries are sufficient for planning, but are not adequate for

dealing with situation independent properties of MPCP domains.

Another major distinction from BATs is the lack of contraposition in defeasible basic action theories. There is on going debate regarding the use of contraposition in defeasible theories (Caminada, 2008). The lack of contraposition in our formalism restricts derivations that can be made from a DBAT to proceed forwards in time. Derivations backwards in time are not supported without additional axioms. This safeguards against problematic derivations that introduce additional, unwanted contradictions. Consider the following example:

**Example 9.** *Consider the following successor state axioms concerning the fluent* $Light(s)$*, grounded for* $S_1 = do(switchOn, S_0)$*.*

$$Light(S_1) \prec Plugged(S); Light(S)$$
$$Light(S_1) \prec Electricity(S); Light(S)$$
$$\sim Light(S_1) \prec \sim Light(S), not\, Plugged(S)$$
$$\sim Light(S_1) \prec \sim Light(S), not\, Electricity(S)$$

*Assume that:*

$$\{Electricity(S), \sim Plugged(S), \sim Light(S)\} \subseteq \mathcal{D}_{S_1}, \text{ and that}$$
$$\{\sim Electricity(S), Plugged(S), Light(S)\} \cup \mathcal{D}_{S_1} = \emptyset.$$

*Without contraposition, there is no ambiguity with respect to* $Plugged(S)$ *in the initial state beliefs. If the contraposition of these rules is considered, this no longer holds. Using the third axiom, and the belief* $\sim Light(S)$*, we can derive* $\sim Light(S_1)$*. Then, using the contraposition of the second axiom we can derive* $\sim Electricity(S)$ *as a side-effect.*

By considering a relation without contraposition and not including the contraposition of the axioms, we simplify the problem and avoid these problematic cases at the expense of disabling reasoning backwards in time.

#### 6.1.2.3 Argumentation-Based Practical Reasoning

The main distinction between our approach and work on argumentation-based practical reasoning is the focus on planning rather than deliberation. Therefore, our formalism is more expressive with respect to the representation of notions related to automated planning as it can represent plans as sequences of actions, rather than just monolithic

entities. In addition, it allows standard features from automated planning such as variables and conditional effects. The explicit representation of actions allow the aggregation of conclusions made from different specifications. On the contrary, approaches that represent the agents' domain knowledge as state transition systems do not clarify how agents can aggregate opinions and argue about specific effects of actions, without merely objecting to a specific state transition altogether.

Formalisms for argumentation-based practical reasoning allow the representation of notions related to deliberation, such as desires and intentions which are beyond the scope of our work. However, since we based our framework on defeasible logic programming, we could combine our theories with other DeLP frameworks for reasoning about intentions and desires (Rotstein et al., 2007, 2008), in order to enable agent deliberation about desirable, rather than just executable, situations. This can be done without modifications to our inference mechanism.

#### 6.1.2.4 Defeasible Argumentation in Planning

The main distinction of our formalism compared to García et al. (2008), García et al. (2007) and Simari et al. (2004) is that we do not assume that initial state beliefs and action specifications are non-contradictory. Similar to their work, our formalism can represent defeasible ramifications, but our planning language is more expressive, allowing variables and conditional effects.

Another important difference is that our approach allows the specification of deductive arguments which contain beliefs that refer to different situations. Therefore, it allows deductive argumentation about the applicability and effects of sequences of actions. On the contrary, in DeLP-POP argumentation is limited to individual states.

In order to reason about conditional effects, the ambiguity propagating nature of defeasible derivation is essential. This leads to correct conclusions regarding conditions that can be derived but are not warranted. If argumentation is performed in a state-by-state manner, conditions that are not warranted are disregarded. In this case, unless the state transition function is adapted, the erroneous assumption that the condition does not hold is made. The distinction between the ambiguity-propagating nature of defeasible derivations with the ambiguity-blocking nature of the classical state transition function was discussed in Section 3.4.2 in detail.

#### 6.1.2.5 Argumentation-Based Reasoning about Dynamic Domains

The differences between our formalism and approaches that utilise argumentation-based reasoning to overcome frame and ramification problems (Kakas et al., 1999, 2001; Vo and Foo, 2005), are primarily due to the different focus, but also to employing different logical formalisms and argumentation frameworks.

Our work focuses on ambiguity introduced by the different viewpoints of the agents with respect to a collectively acceptable outcome. Our formalism is based on defeasible rules given in as successor state axioms, which encode the different views of not only what changes, but also what remains the same.

Our design choice to employ successor state axioms instead of a purely defeasible representation based on default persistence aims to reduce the size of the theory (in terms of predicates and axioms). Successor state axioms allow the representation of every condition that is relevant to a fluent within the body of the axiom that is written for this fluent. This results in succinct representations, since for every fluent literal we write one successor state axiom.

MPCP planning problems contain the planning knowledge of multiple agents. For every agent, we need to encode both effect and frame information. This results in a single axiom per fluent literal (since the resulting axioms for multiple agents can be combined). The frame conditions are important since they allow the agents to argue about what they believe that stays the same, in addition to what they believe is affected by there action. In order to obtain equivalent results from a purely defeasible theory based on default persistence, additional axioms are required. Consider encoding action effects in the following form:

$$\Gamma_L(do(a,s)) \prec\!\!- \gamma_L(s)$$

$\Gamma_L(do(a,s))$ is an auxiliary predicate symbol representing that we have reason to believe that $L$ is an effect of action $a$ in situation $do(a,s)$. The abbreviation $\gamma_L(s)$ denotes the conditions leading to the belief that action $a$ produces $L$ in the successor state. For every literal at least one effect axiom is necessary for the axiomatisation of the domain. Therefore, one auxiliary predicate is necessary for every literal.

Successor state axioms can be encoded in this case using the default negation symbol, without enumerating all relevant conditions, as follows:

$$L(do(a,s)) \prec\!\!- \Gamma_L(do(a,s)); L(s), not\, \Gamma_{\overline{L}(do(a,s))}$$

Here, $\Gamma_{\overline{L}(do(a,s))}$ is an auxiliary predicate denoting that we have reasons to believe that the logical complement of $L$ is the effect of the latest action.

The above axioms are sufficient to encode single-agent reasoning, when each agent has non-contradictory knowledge about the effects of their actions. Moreover, they offer the advantage that frame axioms do not need to be stated explicitly. However, they are not adequate to reason about MPCP problems. In order to reason with the collective beliefs of multiple agents, we need to encode rules that capture their disagreement with respect to frame axioms as well as effect axioms. In order to voice such disagreements, it is necessary to encode axioms stating that the conditions necessary for triggering the effects of an action are not applicable. One way of accomplishing this is to use a different auxiliary predicate for every agent and every fluent. Consider the following rules:

$$\begin{aligned} \Gamma^i_L(do(a,s)) &\multimap \gamma_L(s)\,, \\ \Gamma_L(do(a,s)) &\multimap \Gamma^i_L(do(a,s)) \text{ and} \\ \sim\Gamma_L(do(a,s)) &\multimap \mathit{not}\,\Gamma^i_L(do(a,s))\,. \end{aligned}$$

The auxiliary predicate $\Gamma^i_L$ denotes the belief of agent $i$ that an $L$ is produced in situation $do(a,s)$ as an effect of action $a$.

An alternative way that does not require the additional auxiliary predicates involves explicit rules describing the frame conditions.

$$\sim\Gamma_L(do(a,s)) \multimap \mathit{not}(\gamma_L(s))\,.$$

Encoding the domain in this manner increases the overall number of axioms, and introduces additional auxiliary predicates.

Successor state axioms are the combination of the above axioms, and allow the axiomatisation of the planning domain with one axiom for every fluent literal. Furthermore, their structure asserts that one step derivations made using a successor state axiom supported by beliefs regarding a situation lead to conclusions referring to its successor situation $s$. On the contrary, axioms of the above form require an intermediate step for the derivation of beliefs regarding the effect applicability (i.e. $\Gamma_{\overline{L}(do(a,s))}$). Additionally, the successor state axioms' structure asserts that the conditions leading to conclusion about the value of the literal in the axioms head refer to the predecessor situation term from the situation that appears in the head of the axiom. As a result, a one-step derivation made using a successor state axiom always leads from beliefs re-

ferring to a situation $s$ to beliefs in the successor situation. On the contrary, if auxiliary predicates are employed two derivations are necessary for the same conclusions.

#### 6.1.2.6 Planning under Uncertainty

This work focuses on ambiguity caused by contradictory information, rather than uncertainty which denotes absence of information. However, contrary to the presented set-theoretic MPCP representation, defeasible situation calculus can handle both ambiguity and uncertainty. Uncertainty is encoded simply as the absence of information, whereas ambiguity is encoded as support for both negative and positive values of a proposition. Default negation allows this form of representation as conditions are assumed to hold and are evaluated later in the argumentation phase. As a result, the derivation phase leads to believing that all potential outcomes which are based on uncertain conditions hold, without favouring positive or negative conditions by assuming by default that conditions take on a positive or a negative value. This was discussed in more detail in Section 3.4.2. Even though our formalism can represent uncertainty, the development of algorithms for effectively planning with uncertain domains in conformant or contingent ways are beyond the scope of this work. Conformant planning would require different argumentation semantics (as discussed previously in this chapter) to allow reasoning by cases, whereas contingent planning would require the formalism to account for knowledge producing actions.

### 6.1.3 Reasoning Mechanisms

An important benefit of argumentation theory is that it provides concrete semantics to solutions of MPCP problems. In the previous chapters, we have presented methods for planning with MPCP problems and DBATs, shown that these two formalisms provide equivalent inferential results under the initial state completeness assumption, and discussed the correctness of the proposed algorithms that are based on our formalisms.

The centralised methods we presented operate on the union of all agents' beliefs. Accordingly, the produced results take every relevant argument that can be generated from the joint theories of the agents into account. With respect to soundness these mechanisms produce the most accurate results (i.e. the results are sound with respect to the argumentation framework that contains all arguments that can be generated from the union of the agents' beliefs, and all defeats among these arguments).

We presented a series of algorithms for planning using DBATs. These planners

produce sound results, since potential plans are evaluated using the centralised argument evaluation methods. Search for a plan is performed by searching the situation space. This can be done in a directed manner (i.e. depth-first or breadth-first). If this process is conducted exhaustively, for every plan whose length is within a certain bound, the planning process is complete with respect to the predefined bound.

A directed search of the situation space can be performed using the heuristic value of a state as a guide. We described an algorithm that calculates this value based on the defeasible derivation relation and explained that this process emulates the well-known "no delete lists" heuristics in automated planning. If the planner searches the situation space in a greedy, enforced hill climbing manner, completeness is traded for quickly identifying potential solutions. Alternatively, the heuristic value is used to guide an exhaustive best-first search of the situation space. The latter is complete within the bounds of the search, but not as efficient as successful enforced hill climbing.

Based on the presented translation mechanism, set-theoretic representations with complete initial states produce equivalent results to the defeasible logic-based implementation. We described algorithms that use these methods alongside state-of-the-art planners in order to exploit the highly optimised implementations of these planners. Based on exhaustive search of the state space, planning can be performed in a complete fashion, bounded with respect to the length of the plan. However, the exhaustive nature of such search, and the extensive size of the state-space of moderately sized planning domains, makes this process highly impractical. We propose two hybrid approaches, one based on theory revision and one based on greedy hill climbing search. In both cases we cannot guarantee completeness. A revision mechanism that guarantees completeness is impractical since it leads to extensively increasing theory size to encode the results of the argumentation phase. The greedy hill climbing approach is incomplete by nature, but very efficient in producing solutions.

In order to allow for the distribution of planning and argumentation tasks, and to enable the agents to reach conclusions without communicating their entire belief base, we follow an approach based on argumentation-based dialogue. The iterated disputes dialogue protocol enables the agents to argue in a persuasion dialogue fashion. Agents have clear roles, with the agent proposing a plan being the proponent, leaving the role of the opponent to the other party. This protocol aims at reaching agreement. In combination with the confident strategy, it guarantees sound and complete results with respect to the arguments that can be individually generated. In the special case in which agents argue with conflict-free argument sets, iterated disputes produce sound results

with respect to the union of the agents' arguments. This assumption also applies to agents arguing with standard situation calculus basic action theories.

Iterated disputes are restricted to two-agent dialogues and prohibit the exchange of roles during a dispute. Multi-party iterated disputes allow multiple agents to present proposals and challenge their acceptability by introducing arguments both attacking and defending these proposals. This process may lead to longer dialogues than iterated disputes, but if agents follow the confident strategy (as it is specified for multi-party iterated disputes) the protocol enables a stronger notion of soundness. It guarantees sound results with respect to the union of the agents' arguments, regardless of whether the agents' initial argument sets are conflict-free or not.

Iterated disputes operate at the argument level. As a result, their guarantees are sets of arguments, not individual beliefs. In order to reach more accurate conclusions, we introduce inquiry-based iterated disputes which allow agents to generate arguments in a distributed fashion. This protocol, combined with the confident strategy for inquiry-based iterated disputes, guarantees equivalent conclusions to the centralised mechanism.

The centralised methods do not guarantee privacy since they require the communication of every belief in the agents' theories. Iterated disputes require the agents to communicate arguments only if by doing so they affect the final decision. Multi-party iterated disputes are similar, but they also require the agents to advertise potential moves, which leads to the communication of arguments which are relevant to the acceptance of the current proposal, but which may alter the final outcome of the dialogue (i.e. for advertised moves that are not selected).

Inquiry-based iterated disputes require agents to communicate rules that could potentially form relevant derivations, and share all arguments that are related to partial derivations. Search of the derivation (and the argument) space may not be successful. As a result, this protocol may result in the communication of beliefs that are not relevant to concrete plan proposals (or to their acceptance). Still, since argument inquiry is performed only for claims that are relevant to proposals (and their defeating and supporting arguments). Therefore, the protocol safeguards against the communication of beliefs that are completely irrelevant to the formulation and acceptance of proposals (i.e. there is no partial defeasible derivation which includes these beliefs and leads to conclusions relevant to the arguments being evaluated). As a result, inquiry-based iterated disputes offer limited privacy.

## 6.2 Experimental Results

This section investigates the practicality of our framework by analysing its performance in challenging planning domains of considerable size. Since no standard MPCP domains exist, we selected benchmark domains from the International Planning Competition (McDermott, 2000) in order to be able to test the efficiency of our methods in domains of relevance to the automated planning community.

### 6.2.1 Implementation

We have developed several planners that implement the planning algorithms presented in Chapter 4. This section overviews these and outlines their features. The extensive size of the experimentation domains does not allow the use of implementations that are not based on highly optimised planning code. Therefore, the empirical investigation is limited to our hybrid implementations that delegate the search of potential candidate plans to highly optimised, external planners.

#### 6.2.1.1 DBAT Planner

This planner is based on a prototype Prolog implementation of our theory about planning using DBATs. It allows planning with a rich formalism based on DBATs and the use of extended axioms. More specifically, it supports the following types of axioms:

- Defeasible situation-independent axioms
- Defeasible axioms about the initial situation
- Defeasible action precondition axioms
- Defeasible successor state axioms
- Defeasible state constraints and ramifications
- Presumptions representing observations

Search of the state space may be performed in multiple ways following the methods presented in Section 4.2:

- Depth-first search
- Breadth-first search

- Heuristic search

This planner was implemented as a proof-of-concept system for planning with (extended) DBATs. In practice, it can be used for planning and reasoning about simple domains, but is impractical for use with realistic MPCP problems.

#### 6.2.1.2 Heuristic MPCP Planner

The Heuristic MPCP planner is based on hill-climbing and best-first search. It calculates the heuristic quality of a state based on a solution to the relaxed version of the candidate planning problem which disregards the delete lists of operators, in a similar fashion to Hoffmann and Nebel (2001).

The Heuristic MPCP planner is capable of planning with an expressive planning language. It allows the following features:

- Contradicting initial state beliefs and operator specifications
- Conditional effects, ramifications and state constraints
- Preference orderings over beliefs and operator specifications

The Heuristic MPCP planner allows the intersection of argumentation and planning in the following ways:

- Conclusion-based argumentation
- Condition-based argumentation

The conclusion-based argumentation option performs argumentation to establish whether candidate plans are warranted. This process minimises the argumentation overhead, restricting it to candidate plans. On the other hand, condition-based argumentation evaluates the warrant status of both goals and action conditions. The warrant results are used to prune the search space by dealing with inapplicable actions. This option implements executability-based pruning in the spirit of Algorithm 7.

Compared to the DBAT planner, the Heuristic MPCP planner cannot handle uncertainty in the initial state. Uncertainty can be transformed to ambiguity by adding to the initial state a positive and a negative literal for every uncertain atom. These have minimum preference values. This process is necessary in order to obtain correct inferential results, since MPCP problems must follow the initial state completeness assumption.

#### 6.2.1.3 Hybrid Planners

We have implemented the following hybrid planners based on the algorithms of Section 4.3.3. Hybrid planners delegate search of the space of candidate plans to an external planning process. In order to execute this process, they transform the planning knowledge in a format suitable for a standard planner by following Algorithm 12. Candidate plans that are returned by the external planning process are evaluated using the labelling mechanism described by Algorithm 11.

**Classical planner:** This is the simplest implemented hybrid planner. It uses the external process to search for a candidate plan (without considering acceptability-related information at planning time), which it then evaluates. If this plan is warranted, it is returned. Otherwise, the planner fails. This planner is used as a baseline for our algorithms.

**Conformant planner:** This planner constructs a conformant planning problem by interpreting all ambiguity in the initial state as uncertainty, and calls a conformant external planner to solve this problem. The conformant problem is strictly harder than the MPCP problem, since plans must achieve the goal regardless of which version of the contradictory beliefs might turn out to be true. This planner is used as a baseline for our methods.

**IRB Planner:** The iterative revision-based planner implements Algorithm 13. It calls the external planning process and evaluates the returned solution. If the solution is not warranted it revises the input using information obtained from the argumentation process and repeats the process. The planner fails if the external planner does not return a solution.

**GHC Planner:** The hybrid heuristic planner performs directed greedy hill-climbing search of the state space. It calculates the heuristic quality of states using the candidate plan returned by the external planner. This is done by measuring the the number of actions whose applicability is warranted in sequence. This planner implements Algorithm 14.

### 6.2.2 Experimental Design

We present results with three benchmark domains from the International Planning Competition McDermott (2000):

Table 6.1: Size of experimentation problem instances in terms of the number of ground actions in the non-contradictory domain instances

| **Domain** | *Rovers* | *DriverLog* | *Zeno-Travel* |
|---|---|---|---|
| **Actions** | 900 to 6624 | 637 to 792 | 13000 to 30345 |

**Rovers** is a simplification of the NASA Mars Exploration Rover problem. The goal of the planner is to use multiple planetary rovers in order to explore the environment by taking pictures and gathering samples.

**DriverLog** is concerned with the problem of delivering packages. To achieve this, drivers walk between locations and drive trucks along roads.

**Zeno-Travel** is concerned with embarking and disembarking passengers onto aircrafts, which are flying between multiple locations.

For each domain, we experiment with 5 planning problems taken from the International Planning Competition. Table 6.1 summarises their minimum and maximum sizes. For each planning problem, we construct multiple contradictory problem instances with a varying degree of contradiction $c \in [0.0 : 0.5]$ in steps of 0.1. For every level of contradiction $c$ we generate 50 contradictory problem instances for each planning problem to address the main source of non-determinism in our investigation. Altogether, the performance of every competing planner was evaluated in 4500 problem instances.

We introduce ambiguity in the form of contradictory initial state beliefs and contradictory operators. Rate $c$ represents the probability of introducing a contradictory initial state belief or operator specification, tossing a coin for every effect of every operator. The contradictory operators have a randomly selected precondition. Initial state beliefs and operator specifications are assigned random numerical preference values. The selection of random values follows a uniform distribution.

The following planners compete in the experiments:

- Classical planner
- Conformant planner
- IRB planner
- Greedy Planner

We use Fast Forward (FF) Hoffmann and Nebel (2001), a well-understood, efficient planner, as our classical external planner. The Conformant planner uses Conformant-FF (Brafman and Hoffmann, 2004), which is also based on FF.

The DBAT planner and the Heuristic MPCP planner are excluded from the evaluation. The first cannot deal with domains of the size of those taken from the planning competition. The second is capable of solving a limited number of the smallest IPC problems, but cannot compete directly with the other planners. We have provided these implementations only to verify the implementability of our full formalisms. These planners can solve all presented toy domain examples without difficulty.

The Conformant planner is only used in the first part of our experimentation, which involves initial state contradictions, since it cannot handle uncertainty in operator specifications. The comparison to our methods is not a fair one as the external conformant planner must solve a significantly harder problem, since it cannot resolve contradictions. However, since there are no other approaches that can be directly compared to our system, we resorted to the one that solves the most similar problem, given that it is also capable of solving a transformed version of our problem, in which all ambiguity is treated as uncertainty.

In order to ensure termination, we have imposed bounds on planning times, calls to the external planner, and for the GHC planner, on the number of states it can traverse during its state-space search. The time limit for the conformant planner was set to five times the bound of the external planner.

Unfortunately, in the general case, we cannot verify solution existence, and as a result we can only compare the relative effectiveness of the competing planners. To compute whether a solution exists requires, in the worst case, a complete state enumeration, which is infeasible due to extensive size of the state space in MPCP problems.

### 6.2.3 Results

This section presents the results of our empirical evaluation. First we experiment with problem instances which include contradictions in their initial states. The purpose of these experiments is to illustrate the behaviour of our methods in the simplest scenario. Then we report on the results of experimentation with problem instances that contain both contradictory operator specifications and initial state beliefs. For each experiment we report the following metrics:

**Success rates** reflect the amount of problem instances in which each planner synthe-

sised a warranted plan, for all values of $c$. The mean and standard deviation are calculated across the different IPC problem scenarios.

**Times** describe the planning times for the successful instances. We report the mean planning times and standard deviations for every planner and every degree of contradiction $c$. All times shown are in milliseconds.

**Calls** illustrate the number of times our planners made a call to the external planner requesting a candidate plan. Mean and standard deviation values are reported for the calls made by every planner and every degree of contradiction $c$ on successful instances.

We focus on successful instances, since the results regarding times for external planner calls for the failed instances are significantly skewed due to the imposed termination conditions.

#### 6.2.3.1 Rovers – Contradictory Initial States

The first experiment reports on problem instances of the Rovers domain generated for different degrees of initial state contradiction. Table 6.2 shows the success rates of the competing planners.

All non-contradictory problem instances were solved successfully by all planners. The Classical and the Conformant planner managed to solve a very small subset of the contradictory problem instances. On the contrary, IRB and GHC performed significantly better, especially for $c = 0.1$ and $c = 0.2$. Both algorithms are equally capable of resolving initial state contradictions, and as a result they solved exactly the same contradictory problem instances.

Table 6.3 illustrates the planning times for the successful runs. Initial planning times are low for the Classical, the Conformant and the IRB planner. As the contradiction rate increases, the planning times in the Conformant, IRB and GHC increase, whereas the time for the Classical planner remains low (for $c = 0.1$ and $c = 0.2$).

The GHC planner obtains warrant results for the current state, and delegates the task of searching for a candidate plan to the external planner. Literals that are not warranted are removed from the initial state of the problem that is sent to the external planner, in order to increase the possibility of returning a warranted plan. As a result, the higher the degree of contradiction, the more complex the problem the external planner has to solve, since more information may be missing from the initial state. The

Table 6.2: Ratios of successful instances when attempting to synthesise a warranted plan in all problem instances of the Rovers planning domain for variable degrees of initial state contradiction

| Rovers Success Rates – Initial State Contradiction | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | *Classical* | | *Conformant* | | *IRB* | | *GHC* | |
| C Rate | Success Mean | Std. Dev. | Success Mean | Std. Dev. | Success Mean | Std. Dev. | Success Mean | Std. Dev. |
| 0.0 | 100.00 | 0.00 | 100.00 | 0.00 | 100.00 | 0.00 | 100.00 | 0.00 |
| 0.1 | 10.40 | 10.14 | 8.80 | 7.69 | 33.60 | 14.79 | 33.60 | 14.79 |
| 0.2 | 1.20 | 1.10 | 0.00 | 0.00 | 9.60 | 5.18 | 9.60 | 5.18 |
| 0.3 | 0.00 | 0.00 | 0.00 | 0.00 | 0.80 | 1.10 | 0.80 | 1.10 |
| 0.4 | 0.00 | 0.00 | 0.00 | 0.00 | 0.40 | 0.89 | 0.40 | 0.89 |
| 0.5 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |

Table 6.3: Synthesis times (in milliseconds) for warranted plans in all problem instances of the Rovers planning domain for variable degrees of initial state contradiction

| Rovers Planning Times – Initial State Contradiction | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | *Classical* | | *Conformant* | | *IRB* | | *GHC* | |
| C Rate | Time Mean | Std. Dev. | Time Mean | Std. Dev. | Time Mean | Std. Dev. | Time Mean | Std. Dev. |
| 0.0 | 32.76 | 13.17 | 39.69 | 15.42 | 34.55 | 18.65 | 86.55 | 22.11 |
| 0.1 | 27.62 | 7.39 | 52.14 | 22.19 | 62.81 | 38.22 | 88.15 | 52.05 |
| 0.2 | 34.33 | 6.03 | | | 82.04 | 32.50 | 81.13 | 17.49 |
| 0.3 | | | | | 100.00 | 21.21 | 75.50 | 17.68 |
| 0.4 | | | | | 107.00 | 0.00 | 112.00 | 0.00 |
| 0.5 | | | | | | | | |

standard deviation values remain low. Close inspection of the results showed that there was only a very limited number of instances in which a significant amount of time was used (i.e. two outliers with values over 200 and 500ms respectively).

IRB solves the easiest problems faster than GHC. However, it requires additional time for the problems with a higher degree of contradiction. IRB resolves contradictions that are relevant to the plans returned by the external planner. As a result, the external planner may be called several times until a warranted plan is found, or no plan is returned. In the experiments conducted, we also measured number of calls to the external planner. The calls IRB made to the external planner range from 1 to 4, with the highest standard deviation value being 1.41 for $c = 0.3$.

#### 6.2.3.2 DriverLog – Contradictory Initial States

Similar to Rovers, IRB and GHC outperformed the other planners in our evaluation in the DriverLog domain, especially for $c = 0.1$ and $c = 0.2$. The Classical planner solved more cases than the Conformant planner.

Table 6.4: Ratios of successfully synthesised warranted plans in all problem instances of the DriverLog planning domain for variable degrees of initial state contradiction

| DriverLog Success Rates – Initial State Contradiction | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | *Classical* | | *Conformant* | | *IRB* | | *GHC* | |
| C Rate | Success Mean | Std. Dev. | Success Mean | Std. Dev. | Success Mean | Std. Dev. | Success Mean | Std. Dev. |
| 0.0 | 100.00 | 0.00 | 100.00 | 0.00 | 100.00 | 0.00 | 100.00 | 0.00 |
| 0.1 | 40.80 | 6.26 | 22.80 | 6.57 | 55.20 | 5.22 | 55.20 | 5.22 |
| 0.2 | 13.20 | 9.55 | 5.20 | 5.59 | 28.00 | 14.07 | 28.00 | 14.07 |
| 0.3 | 4.80 | 5.59 | 0.80 | 1.10 | 14.40 | 7.27 | 14.40 | 7.27 |
| 0.4 | 1.60 | 2.19 | 0.00 | 0.00 | 5.60 | 3.29 | 5.60 | 3.29 |
| 0.5 | 0.00 | 0.00 | 0.00 | 0.00 | 1.20 | 1.10 | 1.20 | 1.10 |

In the worst case, IRB conducted an average of 3.00 calls to the external planner with a standard deviation of 1.00. As a result, the mean planning time is twice the time of the non-contradictory problem instances (i.e. $c = 0.0$) in the worst case.

Table 6.5: Synthesis times (in milliseconds) for warranted plans in all DriverLog planning problems for variable degrees of initial state contradiction

| DriverLog Planning Times – Initial State Contradiction | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | *Classical* | | *Conformant* | | *IRB* | | *GHC* | |
| C Rate | Time Mean | Std. Dev. | Time Mean | Std. Dev. | Time Mean | Std. Dev. | Time Mean | Std. Dev. |
| 0.0 | 27.89 | 11.33 | 44.28 | 22.46 | 31.39 | 27.95 | 80.54 | 101.96 |
| 0.1 | 22.34 | 6.41 | 37.37 | 26.86 | 32.33 | 19.02 | 62.58 | 14.50 |
| 0.2 | 21.52 | 3.30 | 34.23 | 13.40 | 43.83 | 25.39 | 63.33 | 13.20 |
| 0.3 | 20.50 | 2.11 | 32.50 | 14.85 | 49.53 | 28.28 | 64.86 | 14.23 |
| 0.4 | 25.75 | 7.27 | | | 48.43 | 18.75 | 69.73 | 20.17 |
| 0.5 | | | | | 67.67 | 26.73 | 64.67 | 18.50 |

GHC required higher planning times, which were the same across all levels of $c$, apart from the easiest case, the non-contradictory problems. This high value was skewed by one outlier in which GHC took 1643ms to solve a problem, and which caused the high standard deviation. The average time to solve the same non-contradictory problem in the other 49 experiments was 66.76 with a standard deviation value of 22.33. The same plan was returned in all cases.

#### 6.2.3.3 Zeno-Travel – Contradictory Initial States

Table 6.6 shows that, compared to the other domains we experimented with, Zeno-Travel was the domain in which the highest success rates were obtained. IRB and GHC outperformed the Classical and the Conformant planners. The planning times in the Zeno-Travel domain are shown in Table 6.7. IRB and GHC managed to synthesise warranted plans in a relatively fast manner. In particular, planning times for GHC remained low for the different values of $c$.

With respect to calls to the external planner, IRB required a smaller number of re-planning steps to synthesis a warranted plan in this domain. In the case with the highest contradiction rate, it required 2.07 calls on average, with a standard deviation of 0.80.

Table 6.6: Ratios of successful instances of synthesising a warranted plan in all Zeno-Travel planning problems for variable degrees of initial state contradiction

| Zeno-Travel Success Rates – Initial State Contradiction | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | *Classical* | | *Conformant* | | *IRB* | | *GHC* | |
| C Rate | Success Mean | Std. Dev. | Success Mean | Std. Dev. | Success Mean | Std. Dev. | Success Mean | Std. Dev. |
| 0.0 | 100.00 | 0.00 | 100.00 | 0.00 | 100.00 | 0.00 | 100.00 | 0.00 |
| 0.1 | 54.00 | 12.00 | 56.80 | 7.82 | 73.20 | 8.32 | 73.20 | 8.32 |
| 0.2 | 30.00 | 13.78 | 23.20 | 11.88 | 47.20 | 10.26 | 47.20 | 10.26 |
| 0.3 | 13.20 | 8.32 | 6.80 | 3.63 | 36.80 | 7.95 | 36.80 | 7.95 |
| 0.4 | 7.60 | 6.23 | 2.00 | 3.46 | 23.60 | 12.68 | 23.60 | 12.68 |
| 0.5 | 4.00 | 4.69 | 1.20 | 1.79 | 16.00 | 5.48 | 16.00 | 5.48 |

Table 6.7: Synthesis times (in milliseconds) for warranted plans in all Zeno-Travel planning problems for variable degrees of initial state contradiction

| Zeno-Travel Planning Times – Initial State Contradiction | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | *Classical* | | *Conformant* | | *IRB* | | *GHC* | |
| C Rate | Time Mean | Std. Dev. | Time Mean | Std. Dev. | Time Mean | Std. Dev. | Time Mean | Std. Dev. |
| 0.0 | 25.54 | 38.43 | 27.49 | 34.79 | 25.80 | 23.33 | 72.40 | 88.54 |
| 0.1 | 22.78 | 5.31 | 27.70 | 5.84 | 29.11 | 13.76 | 64.72 | 9.29 |
| 0.2 | 21.79 | 4.33 | 28.05 | 9.78 | 31.50 | 15.47 | 62.74 | 12.74 |
| 0.3 | 21.76 | 2.66 | 30.06 | 5.03 | 42.78 | 20.98 | 62.82 | 9.89 |
| 0.4 | 21.84 | 2.79 | 29.80 | 5.26 | 39.56 | 16.45 | 61.03 | 5.50 |
| 0.5 | 20.50 | 1.51 | 27.67 | 0.58 | 48.23 | 20.74 | 63.93 | 6.94 |

#### 6.2.3.4 Rovers – Contradictory Initial States and Operator Specifications

The second set of our experiments describes the behaviour of the competing planners in domains with contradictory initial states and planning operator specifications. Table 6.8 reports on the success rates of the planners in the Rovers domain.

Table 6.8: Ratios of successfully synthesising warranted plans in all problem instances of the Rovers planning domain for variable degrees of contradiction

| Rovers Success Rates – Contradiction | | | | | | |
|---|---|---|---|---|---|---|
| | *Classical* | | *IRB* | | *GHC* | |
| C Rate | Success Mean | Std. Dev. | Success Mean | Std. Dev. | Success Mean | Std. Dev. |
| 0.0 | 100.00 | 0.00 | 100.00 | 0.00 | 100.00 | 0.00 |
| 0.1 | 8.40 | 6.69 | 33.60 | 11.61 | 35.60 | 11.08 |
| 0.2 | 1.20 | 1.10 | 7.60 | 4.77 | 8.80 | 4.60 |
| 0.3 | 0.00 | 0.00 | 3.20 | 5.02 | 5.20 | 5.22 |
| 0.4 | 0.00 | 0.00 | 0.00 | 0.00 | 0.80 | 1.10 |
| 0.5 | 0.00 | 0.00 | 0.00 | 0.00 | 0.40 | 0.89 |

The performance of the Classical planner decreased compared to the previous sets of experiments. For $c = 0.1$, it solved 8.4% of the instances on average, and for $c = 0.2$, it solved 1.2% of the instances. IRB and GHC achieved high success rates for problems with $c = 0.1$. However, as the level of contradiction increased, the success rates dropped significantly. GHC slightly outperformed IRB, and was capable of synthesising a limited number of warrant plans in cases with high contradiction rates (i.e. $c = 0.4$ and $c = 0.5$).

Table 6.9 shows the planning times for synthesising warranted plans. GHC required significantly higher planning times. With respect to the degree of contradiction imposed on the problem instances, the planning times of GHC followed an exponential increase. The results for GHC showed high standard deviation values. This was caused by outliers: For $c = 0.1$, without five outlier values 17555, 1150, 4432, 2010 and 3749ms, average times were 90.80ms with a standard deviation of 20.80. Equivalently for $c = 0.2$, after removing the outlier values of 3502 and 1405ms, the average

Table 6.9: Synthesis times (in milliseconds) for warranted plans in the Rovers planning domain for variable degrees of contradiction

| Rovers Planning Times – Contradiction | | | | | | |
|---|---|---|---|---|---|---|
| | *Classical* | | *IRB* | | *GHC* | |
| C Rate | Time Mean | Std. Dev. | Time Mean | Std. Dev. | Time Mean | Std. Dev. |
| 0.0 | 31.64 | 7.20 | 34.88 | 30.54 | 82.72 | 26.97 |
| 0.1 | 29.48 | 4.77 | 88.45 | 69.83 | 411.40 | 1946.08 |
| 0.2 | 31.33 | 4.51 | 99.95 | 76.18 | 304.95 | 767.57 |
| 0.3 | | | 158.50 | 136.72 | 3916.77 | 7252.70 |
| 0.4 | | | | | 12178.00 | 1107.33 |
| 0.5 | | | | | 14382.00 | 0.00 |

Table 6.10: Calls of the external planner in the Rovers planning domain for variable degrees of contradiction

| Rovers Planner Calls – Contradiction | | | | |
|---|---|---|---|---|
| | *IRB* | | *GHC* | |
| C Rate | Calls Mean | Std. Dev. | Calls Mean | Std. Dev. |
| 0.0 | 1.00 | 0.00 | 1.00 | 0.00 |
| 0.1 | 2.52 | 1.38 | 5.70 | 24.73 |
| 0.2 | 2.95 | 1.68 | 4.86 | 13.75 |
| 0.3 | 4.63 | 3.25 | 47.38 | 88.14 |
| 0.4 | | | 202.00 | 0.00 |
| 0.5 | | | 202.00 | 0.00 |

times were 90.00ms with a standard deviation of 14.89. The same holds for $c = 0.3$. In this case, without the outlier values 15895, 17440 and 16554ms, the average planning time is 103.80ms with a standard deviation value of 25.60.

Table 6.10 summarises information regarding the number of external planner calls that were made by each planner. The greedy search performed by GHC resulted in a high number of calls. This way GHC managed to solve some very difficult cases by extensive search of the space of candidate plans returned by the external planner. This also explains the high standard deviation over the times required to synthesise a warranted plan.

#### 6.2.3.5 DriverLog – Contradictory Initial States and Operator Specifications

Tables 6.11, 6.12 and 6.13 outline the results for the DriverLog domain with contradiction in the initial state and operator specifications. IRB and GHC preformed significantly better than the competing planners.

Table 6.11: Ratios of successful instances of synthesising a warranted plan in the DriverLog planning domain for variable degrees of contradiction

| DriverLog Success Rates – Contradiction | | | | | | |
|---|---|---|---|---|---|---|
| | *Classical* | | *IRB* | | *GHC* | |
| C Rate | Success Mean | Std. Dev. | Success Mean | Std. Dev. | Success Mean | Std. Dev. |
| 0.0 | 100.00 | 0.00 | 100.00 | 0.00 | 100.00 | 0.00 |
| 0.1 | 30.40 | 10.81 | 47.20 | 9.44 | 54.00 | 7.35 |
| 0.2 | 6.40 | 3.29 | 20.00 | 6.32 | 37.20 | 8.90 |
| 0.3 | 1.60 | 2.19 | 7.20 | 4.60 | 24.00 | 7.07 |
| 0.4 | 3.20 | 1.10 | 5.20 | 2.68 | 19.20 | 6.57 |
| 0.5 | 0.00 | 0.00 | 0.80 | 1.10 | 10.40 | 4.77 |

GHC was more effective than IRB as it was capable of synthesising plans in more complicated cases for $c = 0.3$, $c = 0.4$ and $c = 0.5$. To achieve this performance, it required higher planning times for difficult problem instances. The high standard deviation values are caused by such cases. This is depicted in Table 6.14. The final

Table 6.12: Synthesis times (in milliseconds) for warranted plans in the DriverLog planning domain for variable degrees of contradiction

| DriverLog Planning Times – Contradiction | | | | | | |
|---|---|---|---|---|---|---|
| | *Classical* | | *IRB* | | *GHC* | |
| C Rate | Time Mean | Std. Dev. | Time Mean | Std. Dev. | Time Mean | Std. Dev. |
| 0.0 | 25.39 | 10.23 | 28.61 | 17.40 | 69.36 | 23.16 |
| 0.1 | 109.66 | 528.91 | 530.45 | 3421.16 | 721.11 | 2450.04 |
| 0.2 | 24.81 | 4.74 | 256.30 | 751.29 | 2934.44 | 8527.61 |
| 0.3 | 23.50 | 1.00 | 65.17 | 30.57 | 2476.47 | 4637.36 |
| 0.4 | 26.25 | 2.49 | 42.85 | 24.22 | 8004.96 | 33168.12 |
| 0.5 | | | 44.00 | 19.80 | 5130.23 | 7238.99 |

Table 6.13: Calls to the external planner in the DriverLog planning domain for variable degrees of contradiction

| DriverLog Planner Calls – Contradiction | | | | |
|---|---|---|---|---|
| | *IRB* | | *GHC* | |
| C Rate | Calls Mean | Std. Dev. | Calls Mean | Std. Dev. |
| 0.0 | 1.00 | 0.00 | 1.00 | 0.00 |
| 0.1 | 1.53 | 0.79 | 8.94 | 35.34 |
| 0.2 | 2.22 | 1.15 | 26.22 | 64.45 |
| 0.3 | 2.17 | 0.92 | 40.18 | 74.88 |
| 0.4 | 1.54 | 0.97 | 55.04 | 89.65 |
| 0.5 | 1.50 | 0.71 | 68.19 | 94.90 |

Table 6.14: Rates of successful problems that were solved quickly by the GHC in the DriverLog domain

| C | Cases | Time Range (ms) |
|---|---|---|
| 0.1 | 91% | 52 to 177 |
| 0.2 | 77% | 59 to 184 |
| 0.3 | 75% | 61 to 179 |
| 0.4 | 71% | 63 to 189 |
| 0.5 | 64% | 73 to 94 |

case (i.e. $c = 0.5$) is the most interesting. Planning times come in two partitions, with 64% between 73 and 94ms and 36% between 10273 and 17194ms.

The planning times for IRB are lower. However, similar to the Classical planner, these values deviate considerably for $c = 0.1$ and $c = 0.2$. This can be attributed to problematic instances that required high planning times from the external planner, since the number of calls to the external planner remained low (i.e. the maximum number of calls was 5). For $c = 0.1$, apart form 115 instances that required between 19 and 397ms, there were three outliers which needed 4644, 21484 and 30313 milliseconds respectively. In a similar fashion, for $c = 0.2$, IRB solved 47 easy instances in between 47 and 261ms, but it was also successful in three hard problems which required a planning time of 2099 to 3781ms.

IRB is more efficient than GHC in the hard problem instances it can solve because it does not manually search the state space. On the contrary, GHC searches the neighbourhood of the current state when the application of an action in the candidate plan is not warranted. Until a good alternative is discovered, GHC non-deterministically selects a state in the neighbourhood, calculates the warrant status of the literals contained in this state, and calls the external planner to generate the corresponding candidate plan.

#### 6.2.3.6 Zeno-Travel – Contradictory Initial States and Operator Specifications

Tables 6.15, 6.16 and 6.17 illustrate the results of the experiments with contradictory instances of the Zeno-Travel domain. The success results shown in Table 6.15 show that GHC performed fairly well, and outperformed IRB. This is evident in the final scenario with $c = 0.5$.

Table 6.15: Ratios of successful instances of synthesising a warranted plan in the Zeno-Travel planning domain for variable degrees of contradiction

| Zeno-Travel Success Rates – Contradiction | | | | | | |
|---|---|---|---|---|---|---|
| | *Classical* | | *IRB* | | *GHC* | |
| C Rate | Success Mean | Std. Dev. | Success Mean | Std. Dev. | Success Mean | Std. Dev. |
| 0.0 | 100.00 | 0.00 | 100.00 | 0.00 | 100.00 | 0.00 |
| 0.1 | 40.00 | 6.78 | 63.20 | 9.65 | 72.80 | 6.72 |
| 0.2 | 20.40 | 7.40 | 34.40 | 8.17 | 48.40 | 6.23 |
| 0.3 | 14.40 | 8.65 | 20.80 | 7.69 | 40.40 | 8.88 |
| 0.4 | 8.80 | 6.42 | 12.80 | 9.96 | 36.00 | 10.58 |
| 0.5 | 5.60 | 2.61 | 9.20 | 4.82 | 28.80 | 13.31 |

Table 6.16: Synthesis times (in milliseconds) for warranted plans in the Zeno-Travel planning domain for variable degrees of contradiction

| Zeno-Travel Planning Times – Contradiction | | | | | | |
|---|---|---|---|---|---|---|
| | *Classical* | | *IRB* | | *GHC* | |
| C Rate | Time Mean | Std. Dev. | Time Mean | Std. Dev. | Time Mean | Std. Dev. |
| 0.0 | 22.52 | 7.88 | 25.79 | 45.66 | 63.95 | 16.85 |
| 0.1 | 33.81 | 17.70 | 87.32 | 147.46 | 1349.73 | 5471.76 |
| 0.2 | 40.61 | 28.07 | 137.88 | 202.20 | 4487.21 | 14908.46 |
| 0.3 | 55.03 | 58.77 | 157.44 | 158.49 | 6976.81 | 19692.22 |
| 0.4 | 58.00 | 30.16 | 160.62 | 145.29 | 6663.92 | 13312.57 |
| 0.5 | 64.07 | 46.20 | 239.39 | 239.51 | 8528.06 | 19643.03 |

Table 6.17: Calls of the external planner in the Zeno-Travel planning domain for variable degrees of contradiction

| Zeno-Travel Planner Calls | | | | |
|---|---|---|---|---|
| | *IRB* | | *GHC* | |
| C Rate | Calls Mean | Std. Dev. | Calls Mean | Std. Dev. |
| 0.0 | 1.00 | 0.00 | 1.00 | 0.00 |
| 0.1 | 1.58 | 0.89 | 13.15 | 48.03 |
| 0.2 | 1.76 | 1.24 | 25.92 | 66.51 |
| 0.3 | 1.85 | 1.45 | 37.62 | 77.35 |
| 0.4 | 1.81 | 1.64 | 52.37 | 88.16 |
| 0.5 | 2.17 | 1.97 | 40.08 | 80.11 |

Table 6.18: Rates of successful instances that were solved quickly by the GHC in the Zeno-Travel domain

| C | Cases | Time Range (ms) |
|---|---|---|
| 0.1 | 89% | 50 to 265 |
| 0.2 | 80% | 56 to 247 |
| 0.3 | 74% | 60 to 353 |
| 0.4 | 62% | 71 to 386 |
| 0.5 | 70% | 78 to 375 |

Increased performance comes at the price of higher planning times. This is shown in Table 6.16. GHC requires significantly higher planning times than IRB. The high standard deviation values for these results show that GHC successfully solved some easy problems, but at the same time, solved a lot of complicated scenarios which required additional effort. This is described in more detail by Table 6.18, which focuses on the planning times required for GHC in easy problem instances. The difficult problems faced by GHC in this domain caused high planning times. Closer inspection of the results showed that, for contradiction levels 0.1 to 0.5, GHC solved numerous (i.e. 6.5%, 12%, 16%, 29% and 19% respectively for different levels of $c$) hard problem instances (which required over 10000ms to solve).

### 6.2.4 Discussion

The GHC planner has the highest success rate in most of the experiments we performed. IRB fails to find a plan in certain cases due to the incomplete nature of the revision process. The increased success rate of GHC comes at the cost of increased planning times, especially for problem instances with higher degrees of contradiction.

The difference in success rates between GHC, IRB, and the Classical planner illustrates the value of resolving the contradictions identified through the planning process. The Classical planner solves more instances than the Conformant planner, due to the highly constrained nature of the conformant planning problem. The classical planner succeeds by accidentally selecting warranted beliefs in the face of ambiguity.

An advantage of IRB is that it only resolves contradictions that are directly related to candidate plans. This is preferable in situations in which argumentation is costly. GHC follows a different strategy, and tries to pass the most accurate state it can evaluate on to the external planner. This way it increases the possibility of receiving candidate plans of higher quality. The warrant evaluation process has been optimised to reuse results from previous states, and was conducted very quickly in practice.

Experimentation with the GHC planner highlights the main difference between MPCP and classical planning: Because of the preference ordering over beliefs in every state, states containing the same literals are not conceptually equivalent – this increases the state space dramatically.

The success of our methods varied across domains. Increased success rates can be attributed to domain-specific elements such as the existence of multiple plans achieving the same goals in the respective domains and plans relying on a smaller number of con-

Table 6.19: Maximum size of returned plans for all competing planners in every domain

| Planners | Classical | Conformant | IRB | GHC |
|---|---|---|---|---|
| Rovers Initial Contradiction | 38 | 38 | 39 | 40 |
| DriverLog Initial Contradiction | 23 | 61 | 34 | 34 |
| Zeno-Travel Initial Contradiction | 24 | 27 | 27 | 27 |
| Rovers Contradiction | 38 | | 40 | 40 |
| DriverLog Contradiction | 23 | | 25 | 27 |
| Zeno-Travel Contradiction | 24 | | 27 | 27 |

ditions. The smaller the number of potential plans that exist in the non-contradictory version of the problem, the higher the probability of introducing ambiguity that renders these plans unwarranted. Accordingly, the more conditions are relevant to the execution of the plan, the higher the probability of introducing ambiguity on important conditions.

Contradictions alter the structure of planning problems, especially when $c$ takes on high values. This affects the number of actions that must be executed to solve these problems. Our approach does not focus on returning plans of minimum size, nor does the external planner guarantee that the smallest candidate plan is always returned. Regardless, the sizes of the returned plans were in most cases comparable. Table 6.19 shows the maximum size of plans returned by the competing planners for every set of experiments.

Our results illustrate that the problem of planning with contradictory theories is very hard. The introduction of contradictory planning operators significantly increases the size of the planning domain. Additional actions are instantiated, introducing more options that the planner may consider in every step. There were cases in which the number of actions was tripled. This, in combination with the non-standard form of transformed planning theory (which is necessary to hide the contradictions from the planner), may result in problems that the external planner cannot handle as efficiently as hand-coded instances.

Our planners were capable of synthesising plans in extensive contradictory theories outperforming simple approaches that do not utilise information related to the acceptability of arguments. In most cases, our planners synthesised warranted plans in an efficient manner. GHC required significant time to solve the most complicated

cases. IRB performed slightly worse, but maintained planning times comparable to the simple baseline approaches.

## 6.3 Applicability in Real World Domains

In order to discuss the value of MPCP in a broader context, we describe instances of MPCP problems that appear in important real-world domains. The purpose of this section is to present examples to illustrate the commonality of the problem, and discuss the solution that can be provided by our methods. In addition, our analysis highlights interesting domain characteristics and how these relate to the suitability of our methods. Examples of such characteristics are privacy concerns, authority, individual perspective, safety critical applications, and time constraints.

### 6.3.1 Emergency Response Domain

The first example is inspired by RoboCup Rescue (Kitano and Tadokoro, 2001). RoboCup Rescue focuses on disaster rescue and emergency decision support. It involves the integration of disaster information, prediction, planning, and human interfaces in a virtual world struck with disasters, such as earthquakes and fire. Mobile and static agents must cooperate in order to provide rescue. The following agent types are supported:

- Mobile agents: civilian, fire-fighter, rescuer, police
- Static agents: fire station, police station, hospital, refuge

Emergency response agents are cooperative. Even though they may have individual goals they are pursuing, they must not exhibit individualistic, strategic behaviour, since they collectively need to work towards the overall welfare of the system, that is rescuing civilians from disaster-struck parts of the environment.

The RoboCup Rescue domains are large and complex, and as a result agents have partial views of the world. Due to the unexpected disasters and the actions performed by other agents, agents may hold outdated, erroneous beliefs. In such settings, agent action must be prompt and directed. Sharing all agents' views is not practical since it involves the communication of extensive amounts of detailed information which may be irrelevant to the operations specific agents engage in. In addition, due to the extensive size of the domain, a centralised planning approach based on a joint planning

theory may be impractical. Even though such a theory would lead to correct decisions, it may be impossible to make these decisions in reasonable time.

Since agents operate in an unstable environment, and cannot observe every action taken by other parties, their observations are defeasible. The confidence level of an observation is relative to how outdated this information is. Additionally, since observations are made by different sensors, credibility values may be discounted based on confidence information on the quality of sensors, or the proximity of agents to the to the location of the observed events.

The following example illustrates specific MPCP problem instances inspired by the RoboCup Rescue domain.

**Example 10.** *An ambulance agent synthesises a plan for reaching and treating civilians that have been evacuated from a collapsed building. This plan involves moving to the location of the building, offering aid to the injured civilians and transporting them to the closest hospital. Assume that this agent has not been operating in this specific area after the earthquake. As a result, the agent's beliefs regarding which routes remain unblocked are outdated. Also, assume the police centre is responsible for the collection of information regarding blocked roads in this area from mobile agents.*

The agent can communicate its transportation plans to the police centre. The centre, after revising the plan, can identify actions that cannot be executed and communicate the reasons to the ambulance agent. The ambulance agent re-plans based on this information and communicates the new plan. This plan is then accepted by the centre agent.

This is an example of a two-person iterated dispute among the agents in the emergency response domain. The different views of the agents in this case are the result of divergent spatial information. The beliefs of the central agent have higher credibility since this agent is responsible for collecting the most up-to-date information regarding the specific area of the environment. The proposal arguments in this example are minimal, with the ambulance agent communicating only the transportation route, without getting into detail regarding the reasons behind the belief that this route is valid.

In a more elaborate example, the central agent may re-plan, introducing additional actions executed from a fire-fighter agent to clear the route from debris, allowing the ambulance agent to proceed. In this case, the plan must be communicated to the fire centre agent as well, in order to organise the collaborations among the different fire-fighter agents under its command.

**Example 11.** *Assume the specifications of the actions the agents apply to the environment differ across agents of different roles and authority. For example, the role of fire station agents is to coordinate the firefighter agents under their command. Their position is static. On the other hand, the firefighter agents operate in the environment, and affect it by actions such as extinguishing fires and clearing debris. Firefighters need to hold very detailed specifications of the actions they execute. For instance, extinguishing a fire may involve conditions related to specific details of the site and characteristics of the fire. Such low-level details cannot be observed by the fire station, and are irrelevant to their role. Accordingly, assume that they hold high-level specifications of the actions that are executed by mobile units. Following this example, the action of a firefighter extinguishing a fire may only require that the location of the firefighter is approximate to the location of the fire.*

In this scenario, fire station agents can use their generic specifications to construct plans coordinating the agents, and communicate these to the relevant agent by initiating appropriate dialogues. These agents then evaluate the plans, with respect to their detailed operator specifications and local views, and raise possible objections. Every objection is related to incompleteness in the high-level operator specifications held by the station agents and lack of low-level observations. In this case, the detailed specifications and observations of the mobile agents must be assigned higher preference values than the high-level ones held by the disembodied agent. The dialogue process serves as the medium to align the agents' beliefs that are relevant to the specific task, while ensuring that mobile agents do not follow plans that are ineffective according to their beliefs.

Incorrect or incomplete views of the environment may cause plans to fail. The situation is harder if we consider exogenous events, such as fire spreading and buildings collapsing, occurring unexpectedly and interfering with the agents' actions. In this case, the re-planning process has to rely both on observations made during plan execution and the anticipated outcomes of the executed actions when no relevant observations have been made.

**Example 12.** *Assume the agents agree on a joint plan which fails during execution. Agents have individually collected observations during plan execution. In order to achieve their goal they must re-plan, while taking into account their observations, which are distributed, incomplete and potentially contradictory.*

Following the multi-party protocol, an agent may initiate the re-planning process by presenting a potential plan, leading from the current situation to a situation in which the goal is achieved. In order to identify what holds in the current situation the agent relies on the observations made during the execution of the plan and the expected outcome of actions, as described by the agent's operator specification. Beliefs that correspond to recent observations have higher preference over conclusions regarding the same literals derived using the specification of the operators and other potentially outdated beliefs.

If the plan is safety-critical the agents may attempt to collaboratively generate support for potential plans using the inquiry dialogue protocol. This process is expensive, but enables the agents to utilise their collective observations related to the situations in which the plan was executed and exploit their full combined knowledge.

### 6.3.2 Medical Domain

While medical knowledge increases in size and complexity, there is an increasing need for computational mechanisms that can be used by medical practitioners (Fox and Das, 2000). Recent research proposes argumentation theory as a good paradigm for decision support in medical applications (Fox et al., 2007).

Consider the problem of construction and execution of treatment plans. Decisions must be made by practitioners with different specialisations and levels of training. This problem is well-suited to our approach: Patient treatment usually requires multiple actions that are interdependent. There is extensive knowledge encoding the anticipated results of such actions. Due to its extensive size and complexity, as well as its constant development, medical theories may be incomplete and potentially contradictory. Usually, real-world decision making and execution involves the cooperation of several practitioners trained in different specialisations. The decision-making process is safety-critical. Ideally, the selected treatment plans must be defendable against all possible objections. In addition to this, privacy is important when decision making involves confidential personal information.

**Example 13.** *Consider the decision making process between practitioners seeking agreement on a patient's treatment plan. Assume practitioner* D *is the patient's personal physician, holding the patient's complete history records. An additional specialist* S *is participating to propose potential treatments which exceed* D*'s specialisation. The patient is also involved in the decision-making process when potential side-effects*

*are identified which might affect their willingness to seek a therapy which is associated with such risks.*

Our framework can be employed as a mechanism for implementing this process. Iterated disputes allow participating agents to propose potential solutions, and other agents to evaluate them. In this case, the specialist agent could propose potential solutions, based on initial information about the patient and specialised knowledge about potential treatment actions.

By communicating the initial proposal, the specialist enables $D$ to inspect the assumptions under which $S$ presents the treatment and check whether these assumptions hold for the specific patient. $D$ holds specific knowledge about the patient's details, and any objections raised based on the patient's history would be assigned a higher ordering than generic assumptions made by the specialist. The specific details regarding the treatment actions presented by $S$ are assigned higher preference values than the generic views $D$ holds about potential treatments.

Patient involvement in the decision-making process can be realised using our minimal plan proposal approach. After agreement on a potential treatment plan, $D$ communicates the plan to the patient and describes further details after questions are made by $P$ regarding potential negative side-effects. The patient, finally, decides if certain conditions are acceptable and accepts or rejects the presented treatment. Patients' beliefs about their willingness to accept potential side-effects are assigned higher preference than the generic beliefs held by the practitioners representing general patient tendencies towards such side-effects. These beliefs of the patient can be utilised by the practitioners during the construction of further proposals.

Following the previous example, we consider the problem of monitoring the execution of a treatment plan:

**Example 14.** *During the execution of a treatment plan, observations are made evaluating the patient's response to the treatment plan. If these indicate that the plan did not have the anticipated effects, and necessary conditions for the success of future actions were not produced, then execution should stop, since search for an alternative plan is necessary.*

Based on the observations, it is possible to identify whether the plan is expected to have the intended outcomes, or if additional actions are required. Observations can be encoded in our system as beliefs regarding intermediate situations which are assigned high preference orderings. As a result, conclusions reached using observations would

have higher preference than conclusions reached using using the axioms specifying the anticipated effects of actions. For beliefs that are not coupled to observations, it is only possible to reach conclusions regarding their states after the intermediate application of the plan, by only considering the anticipated effects of actions.

### 6.3.3 Distributed Vehicle Monitoring

The following example is inspired by the Distributed Vehicle Monitoring Testbed (Conway et al., 1983), which is concerned with the problem of monitoring and interpreting data from spatially distributed sensors. Consider the following example:

**Example 15.** *A set of agents are spatially distributed in a domain. Their aim is to identify the path of a car that passed from this domain based on observations from their sensors. We assume that different agents are mainly responsible for different areas of the domain. However, there are intersections in their viewpoints. Sensors may be faulty, but their functioning is checked on a regular basis.*

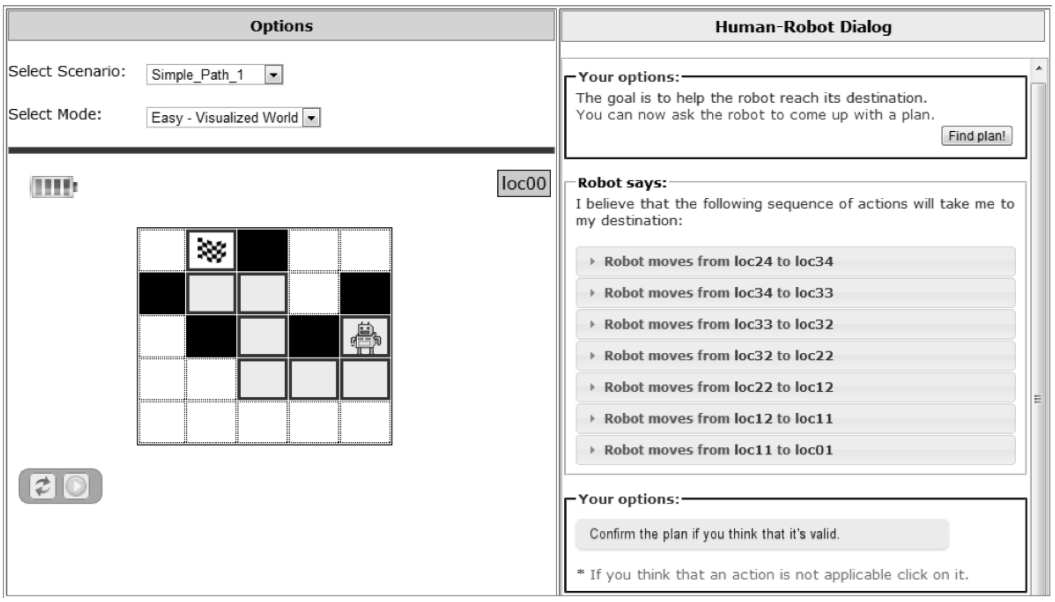
The above problem can be formulated as an MPCP problem. Based on their observations and other assumptions about the initial location of the car, the agents present possible plans the car could potentially have followed to their peers, which correspond to potential paths the car has traversed. Based on these distributed observations the agents evaluate the acceptability of the plans in a distributed fashion.

Knowledge is distributed among the agents since agents hold observations for different locations of the map before the dialogue process. Observations are assigned higher preference values than inferences made based on assumptions regarding what may have happened. Conflicts caused by contradicting observations is resolved by assigning relative preference values to observations based on the quality of their sensors. More specifically, sensors that haven't been checked recently are considered to be less credible and are assigned to lower preference values.

Dissuasion about the potential plans can follow any of the proposed dialogue-based protocols. The selection of which protocol would be more appropriate is based on how the agents are connected, whether the application is safety-critical and whether decisions have to be made in a timely fashion.

The agents can synthesise potential plans by introducing assumptions, which can later be invalidated. These assumptions must have the lowest possible preference. After every dialogue process the agents take a step towards aligning their beliefs by introducing observations made by their peers into their knowledge.

Figure 6.1: Argudem: a human-agent interface for dialogue about MPCP problems




### 6.3.4 Human Computer Interaction

Argumentation is an interesting tool for human-computer interaction. Instead of simply communicating a claim, argument structure enables the implicit description of the process which leads to this conclusion. Morali (2011) empirically investigates the efficiency of argumentation-based interaction with human users in the context of an automated planning-based system called *Argudem*, a human-agent interface built on top of our MPCP set-theoretic implementation. Figure 6.1 shows Argudem's interface.

The Argudem scenario involves a planning agent situated in a grid world which contains obstacles and a goal destination. Through an interface, the human user can perceive the state of the world, which may be different from the view the situated robotic agent has on the environment. The agent synthesises plans to reach the goal destination and communicates these to the user. Due to the different views of the environment, plans may not comply with the user's view of the world. In this case, the human user can initiate a dialogue process in order to persuade the agent about the possible pitfalls of the plan.

Argudem offers a modular interface consisting of a visual as well a descriptive view of the domain, the anticipated effects of the plans to the state of the world, and the arguments exchanged during the dialogue. The modularity of the interface allows Argudem to function in two modes: "world visualisation" and "expert oriented". The first depicts the state of the environment after the application of actions, and the agents

proposed path, using a grid-like representation. The later, in the place of the domain visualisation, offers a tree-like representation of the plan proposal argument presented by the agent.

The following example describes Argudem in use:

**Example 16.** *The robot initially comes up with a plan to reach the goal location. The observer inspects the plan and identifies a problem, that the robot is about to attempt to move through an obstacle. Accordingly, the user raises an objection against the applicability the relevant action in the plan. The agent replies that according to their operator specification of the action all preconditions hold. The user then inspects this specification and challenges the problematic condition, that for instance the condition stating that location*$(2,2)$ *does not contain any obstacles. The robot explains that the belief that the condition holds is justified by an initial state belief. The user corrects the robot by explaining that this belief is not correct. Next, the agent accepts the users view and updates its theory. The dialogue continues with the robot searching for an alternative plan, utilising the human user's input.*

Morali (2011) empirically investigated the educational value of Argudem and the quality of the system in terms of its effectiveness and efficiency through a series of experiments with human users of different backgrounds. The experimentation produced positive results with respect to the quality of the interface and the demonstrator's ability to convey key automated planning and argumentation concepts to novice users.

## 6.4 Summary

In this chapter, we have presented a comprehensive evaluation of our framework. This evaluation focused on three important questions which arise from our hypothesis. Here, we summarise the evaluation results with respect to these questions.

**Is the problem of multi-perspective cooperative planning common?**

We presented a series of significant problems illustrating that the problem of MPCP is evident in different important domains. Different instances of the problem have different important characteristics. For example, some applications are safety-critical, whereas in others agents must reach agreement promptly. In some applications agents have privacy concerns, whereas in others there is a predefined structure of authority.

We have described how these characteristics dictate which of our methods are appropriate in such settings, and how agents can use them to capture the problem domain in terms of a MPCP problem.

Several approaches in the literature are concerned with related problems. More specifically, other approaches in the literature focus on the problem of multiagent agreement on deliberation and planning. This illustrates the commonality of the problem of coordinating agent behaviour in the light of contradictory knowledge.

**Is argumentation theory suitable for the specification of the MPCP problem?**

Argumentation theory, combined with a suitable language which enables reasoning about contradictory dynamic domains, is suitable for the specification of MPCP planning problems. Our proposed formalism is based on defeasible basic action theories in defeasible situation calculus and is strictly more expressive than the initial set-theoretic representation of MPCP. In addition, argumentation theory offers concrete semantics for the specification of plan acceptability, which is essential for providing a rational solution to the problem. An important benefit accrued from this is that it enables the analytical evaluation of the effectiveness our methods.

**What is the quality of the proposed solution to the problem of MPCP?**

Our framework proposes a family of algorithms and protocols for solving the MPCP problem. These mechanisms make different guarantees, which directly correspond to the amount of beliefs and arguments that the agents are required to consider in each case. All methods are sound with respect to their guarantees.

We have evaluated the practicality of our methods with respect to the ability of the formalism to encode domains in succinct representations and the ability of the planning methods to synthesise plans in reasonable time in extensive domains. The proposed formalism allows features such as variables, conditional effects, and in the extended version of our theories state constraints and ramifications. As a result, it offers a highly expressive language for the representation of planning domains. However, this language does not allow other interesting features found in PDDL and situation calculus, such as functions and quantifiers. In addition, our methods are limited to answering bounded-length queries, which are sufficient for planning, but do not handle unground queries regarding properties of the domain or reasoning backwards in time. Finally, compared to argumentation-based approaches on practical reasoning

and deliberation, although our framework is more expressive with respect to encoding planning domains, it does not by default enable reasoning with deliberative notions such as intentions and desires.

We have evaluated the practicality of our approach in planning domains from the International Planning Competition. The extensive size of these domains makes it impossible to rely on a simple argumentation approach. However, by using the inherent characteristics of the planning domain and utilising efficient, state-of-the-art planners our methods managed to synthesise warranted plans in reasonable times.

The main contributions of this chapter are:

- Evaluation of the formalism with respect to expressiveness compared to related approaches.
- Empirical evaluation of the practicality of our methods in extensive domains from the International Planning Competition.
- Discussion of applicability of MPCP in a broader context of relevant applications.

# Chapter 7

# Conclusions

This chapter summarises the research presented in this thesis, and presents the most significant contributions of our work. We propose interesting directions for further research, and provide some concluding remarks.

## 7.1 Thesis Summary

We have presented an argumentation-theoretic approach to the problem of multi-perspective cooperative planning under ontological agreement. This is the problem of synthesising a plan for multiple agents which share a goal but hold different views about the state of the environment and the specification of the actions they can perform to affect it.

The background chapter provided a thorough overview of work in the areas of automated planning, reasoning about action and argumentation. Research related to the MPCP problem was described in further detail, illustrating the novelty of our work and outlining its context.

In Chapter 3 we formally specified the problem of MPCP using a set-theoretic and a defeasible logic formalism. We adapted classical set-theoretic planning notation in order to define the first sub-problem of MPCP, the problem of synthesising candidate plans. A direct result of this formalism is the ability to use standard planning techniques with only minor modifications to their input. This is very important, since modern planning systems are highly optimised to achieve scalable plan synthesis in complex domains. Based on argumentation theory and the novel defeasible situation calculus formalism, we formalised the notion of acceptability and concretised the decision-making sub-problem of MPCP. In order to ensure that the solutions to the

two sub-problems can be combined correctly, we bridged the inferential results of the proposed formalisms.

Based on the suggested representations, Chapter 4 introduced techniques that enable sound reasoning and planning in MPCP domains. In order to increase the efficiency of our methods, we have proposed a series of optimisations that prune the search space based on the inherent structure of the planning domains. In addition, we presented heuristic planning algorithms that exploit the capabilities of off-the-shelf planners to increase the scalability of our approach.

In Chapter 5, we presented a family of dialogue-based protocols that allow agents to search the space of potential solutions to a problem in a distributed fashion, resolve contradictions, and align their beliefs. Each protocol we proposed has different properties, and is suitable for domains with diverse characteristics, as for instance strict time constraints, or safety-critical applications. Our methods terminate and provide guarantees in terms of the correctness of the returned solutions.

Chapter 6 conducted a comprehensive evaluation of our methods. It summarised our analytical results and reported empirical experiments with contradictory instances of benchmark planning problems. The experimentation illustrated that the proposed planning techniques can synthesise plans in reasonable time in extensive contradictory planning domains. Finally, we described the commonality of MPCP, and explained how our methods can be used to tackle such problems, based on examples of MPCP problem instances from scenarios inspired by important real-world problems.

## 7.2 Summary of Contribution

The contributions of this thesis are outlined as follows:

**Formalisation of the MPCP:**

MPCP has been specified using two formalisms: we adapted the classical set-theoretic planning notation, and presented defeasible situation calculus, a novel formalism based on the combination of defeasible logic programming and situation calculus. The first maintains a close relation to classical planning and allows the use of standard planning techniques with only minor modifications, whereas the latter enables the specification of MPCP problems in an elegant way based on deductive argumentation. In order to ensure correctness, we have bridged the inferential results of the two formalisms.

**Practical algorithms for reasoning and planning with MPCP domains:**

We focused on practicality: We proposed heuristics that exploit the inherent structure of the planning domain to prune the search space, adopted powerful planning heuristics and provided algorithms that allow the use of off-the-shelf planning systems.

**Distributed mechanisms for reaching agreement:**

In order to allow the distribution of tasks, we proposed a family of abstract dialogue-based collaborative protocols. Based on different instances of MPCP that appear in important problems, we illustrated different characteristics of MPCP and explained which mechanisms are better suited to provide solutions.

**Analytical and experimental evaluation of our methods:**

We conducted a comprehensive evaluation of the proposed methods. Analytical investigation showed the effectiveness of our approach. Empirical experimentation illustrated that our algorithms can synthesise plans in reasonable times with contradictory instances of benchmark planning problems.

The work conducted for this thesis lies at the intersection of multiagent systems, automated planning, reasoning about dynamic domains and argumentation. Our research contributes to these fields in the following ways:

**Automated Planning:** This work is the first attempt on relaxing the assumption of classical planning that planning knowledge is consistent. In addition, it presents the first implemented system that can synthesise plans in a scalable way when there exist multiple, contradictory views about the planning domain.

**Reasoning about dynamic domains:** Defeasible situation calculus is a novel, expressive formalism for argumentation-based reasoning about contradictory dynamic domains.

**Argumentation:** The proposed algorithms and heuristics allow scalable argumentation in extensive environments in which a naive argumentation approach is infeasible.

**Multiagent Systems:** We present a family of novel argumentation-based dialogue protocols for distributed problem solving.

**Artificial Intelligence:** Apart from addressing an important multiagent planning problem, our work bridges research on planning, reasoning about action and argumentation.

## 7.3 Practical Applicability Context

This section outlines potential uses of the proposed methods within a multiagent system and overviews its strengths and limitations. MPCP has been formalised based on a series of assumptions, which set the scope of our approach. First of all, agents operate under ontological alignment. Agents share the names of propositions and actions, and there is agreement about the concrete entities these names refer to. In addition, agents are assumed to be cooperative, and there exists a set of shared goals that has been identified *a priori*. We assume agreement does not involve strategic considerations regarding how the workload is spread among the agents. Finally, MPCP focuses on classical-style planning involving sequential plans.

Multiagent execution can be represented by reserving a term for every planning operator to encode the agent (or agents for joint actions) that will execute the action. Action capabilities can be also encoded within the specification of planning operators, and agents may hold contradicting specifications about such capabilities as well. Concurrency and durative actions are not inherently supported. Of course, they can be supported within the limits of existing, conventional planning domain transformations.

Our approach is domain independent. Our methods can be applied to any MPCP domain that is represented using the proposed formalisms. The expressive power of these formalisms is comparable to modern planning languages, and exceeds the capabilities of some state-of-the-art planners (e.g. by allowing state constraints and ramifications). Its main limitations compared to more expressive languages are its lack of support for functions, metrics and explicit quantifications.

Our methods can be applied to provide acceptable solutions in planning domains (as for instance the domains from the deterministic track of the IPC), when knowledge about these domains is contradictory, or when different agents have different views about the planning environment. Agents may also hold information regarding the credibility of their individual beliefs. Note that existence of such information is not necessary, and that this information need not be shared. However, credibility values presented by an agent must be accepted by the other parties.

The proposed methods enable the identification of potential objections to concrete

plans (i.e. reasons why these plans will not achieve the goal). Also, if the agents hold rich information regarding the credibility and source of their beliefs, or the mechanics of the planning domain, they can utilise this information to argue about the validity of these objections. By utilising all relevant knowledge and investigating potential objections, plans that are based on incorrect information (with respect to the agents' knowledge) can be disregarded, increasing the quality of accepted solutions.

Depending on the specific characteristics of an MPCP problem different methods can be used among the ones we have presented. The centralised method should be used when communication of all beliefs prior to planning is possible. When communication of all beliefs must be avoided (due to time or privacy constraints) and agents can individually construct potential plans, the iterative dispute protocol can be used. In this version of the protocol, agents do not switch between roles, which leads to reaching a decision faster, but the protocol lacks the ability to safeguard the plan against argumentation paths that can be formed from the union of the arguments that are available to the agents. When multi-party agreement is required (i.e. more than two agents), the agents must argue in pairs. Alternatively, the multi-party version of the protocol may be selected. The main difference in this case is that agents raise arguments both supporting and defeating plans, switching roles when appropriate. This protocol ensures that all arguments that are relevant to a plan proposal, and can be constructed from the beliefs of each agent, will be weighted. When operating in safety-critical domains in which important beliefs are distributed among the agents, the inquiry-based version of the protocol is preferable. This version allows distributed argument generation (including generation of proposal arguments), and achieves results that are equivalent to the centralised method.

## 7.4 Future Work

There are many opportunities for further work within the context of MPCP. We outline the most significant:

**Strategic aspects:** A central assumption of this thesis is that agents are purely cooperative. They all share a common goal, and as a result, they are willing to accept every plan that can be defended against possible objections, regardless of how this plan distributes the effort required for its execution. The general case of multi-perspective planning is not purely cooperative. In a strategic setting,

agents are only willing to collaborate if by joining efforts they can achieve goals that are otherwise unattainable, or would require them to make additional effort. Interesting issues arise in a more competitive setting, not only related to which plans the agents are willing to execute, but also regarding which pieces of information agents are willing to communicate.Sharing information involves strategic decisions, since different common knowledge may lead to reaching agreement on different plans, with a different distribution of labour.

**Domain-specific problems:** The methods presented in this thesis are domain independent. Our formalisms rely on a general representation of the planning environment and our algorithms are optimised based on the general structure of planning domains. Further work can be performed on identifying common domain specific attributes that can be utilised to increase the performance of the proposed methods. These characteristics may reflect particular domains, as for example the emergency response domain, or be related to specific features of the language, as for example specifying conditions using exclusively positive literals.

**Implementation:** Our results illustrated that planning in MPCP domains can be conducted in an efficient manner using state-of-the-art planners. We allowed this by modifying the input of the planner in order to delegate search for candidate plans. We believe that this approach can be further optimised by modifying such planners, producing native, highly-optimised MPCP implementations. The resulting planner would enable a more extensive search of the state space, while better utilising the results of the argumentation process.

**Concrete theories for reasoning about planning beliefs:** In order to resolve contradictions on the belief level we resorted to preference orderings. For the experimentation process, we instantiated these preferences using arbitrary numerical values. An interesting extension is to investigate the ramifications of combining our systems with concrete defeasible logic theories for reasoning about the sources and credibility of beliefs. This way, further work can be conducted on identifying heuristics and strategies for deciding when to argue, in situations in which the resolution of conflicts requires extensive argumentation.

**Expressiveness:** The presented work isolates the problem of MPCP from other interesting aspects of multiagent planning, such as distributed execution and par-

tially ordered plans. Interesting directions of research include the investigation of MPCP with more expressive formalisms allowing representation and reasoning about distributed plans for actions, and partial solutions.

## 7.5 Concluding Remarks

This thesis presented an argumentation-theoretic solution to the problem of multi-perspective cooperative planning under ontological agreement. MPCP relaxes the implicit assumption of classical planning that planning beliefs are consistent, which is not realistic for complex, multiagent domains. The resulting problem is strictly harder than classical planning, since the additional constraint of plan acceptability is imposed on solutions. We separated the two problems and followed a structured approach that exploits the advantages of both modern planning and argumentation techniques. Deductive argumentation allowed the formal specification of the notion of plan warrant based on an expressive logical formalism. Based on this notion, the correctness of our methods was shown. In addition, in order to develop efficient methods, we deviated as little as possible from classical planning. This allowed the delegation of the search of the state space to highly optimised, external planners. The results of our empirical experimentation show that although the MPCP problem is very complex, it is possible to synthesise warranted plans in contradictory instances of benchmark planning problems within reasonable times.

# Bibliography


Allen, J. and Ferguson, G. (1994). Actions and events in interval temporal logic. *Journal of logic and computation*, 4(5):531–579.

Amgoud, L. (2003). A Formal Framework for Handling Conflicting Desires. *Lecture Notes in Computer Science*, 2711:552–563.

Amgoud, L. and Cayrol, C. (1998). On the acceptability of arguments in preference-based argumentation. In Cooper, G. F. and Moral, S., editors, *Proceedings of the 14th Uncertainty in Artificial Intelligence (UAI 1998)*, pages 1–7.

Amgoud, L. and Cayrol, C. (2002). A reasoning model based on the production of acceptable arguments. *Annals of Mathematics and Artificial Intelligence*, 34(1-3):197–215.

Amgoud, L. and Cayrol, C. (2004). On the use of an ATMS for handling conflicting desires. In Dubois, D., Welty, C. A., and Williams, M.-A., editors, *Proceedings of the Ninth International Conference on the Principles of Knowledge Representation and Reasoning (KR 2004)*, pages 194–202.

Amgoud, L., Devred, C., and Lagasquie-Schiex, M. (2008). A constrained argumentation system for practical reasoning. In Padgham, L., Parkes, D. C., Müller, J. P., and Parsons, S., editors, *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, pages 429–436.

Amgoud, L., Devred, C., and Lagasquie-Schiex, M. (2011). Generating possible intentions with constrained argumentation systems. *International Journal of Approximate Reasoning*, 52(9):1363–1391.

Amgoud, L., Maudet, N., and Parsons, S. (2000a). Modelling dialogues using argumentation. In Durfee, E., editor, *Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS 2000)*, pages 31–38. IEEE.

Amgoud, L., Parsons, S., and Perrussel, L. (2000b). An argumentation framework based on contextual preferences. In Cunningham, J., editor, *Proceedings of the International Conference on Formal and Applied and Practical Reasoning (FAPR 2000)*.

Atkinson, K. (2005). *What Should We Do?: Computational Representation of Persuasive Argument in Practical Reasoning*. PhD thesis, University of Liverpool.

Atkinson, K. and Bench-Capon, T. (2007a). Action-based alternating transition systems for arguments about action. In Collins, J., Faratin, P., Parsons, S., Rodríguez-Aguilar, J. A., Sadeh, N. M., Shehory, O., and Sklar, E., editors, *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence (AAAI 2007)*, pages 24–29.

Atkinson, K. and Bench-Capon, T. (2007b). Practical reasoning as presumptive argumentation using action based alternating transition systems. *Artificial Intelligence*, 171(10-15):855–874.

Atkinson, K., Bench-Capon, T., and Mcburney, P. (2005). A dialogue game protocol for multi-agent argument over proposals for action. *Autonomous Agents and Multi-Agent Systems*, 11(2):153–171.

Augusto, J. and Simari, G. (2001). Temporal defeasible reasoning. *Knowledge and Information Systems*, 3(3):287–318.

Baral, C. and Lobo, J. (1997). Defeasible specifications in action theories. In Pollack, M. E., editor, *Proceedings of the Fifteenth international joint conference on Artifical intelligence (IJCAI 1997)*, volume 15, pages 1441–1446.

Baroni, P. and Giacomin, M. (2007). On principle-based evaluation of extension-based argumentation semantics. *Artificial Intelligence*, 171(10-15):675–700.

Baroni, P. and Giacomin, M. (2009). Semantics of abstract argument systems. In Rahwan, I. and Simari, G., editors, *Argumentation in Artificial Intelligence*, pages 25–44. Springer.

Belesiotis, A., Rovatsos, M., and Rahwan, I. (2009). A generative dialogue system for arguing about plans in situation calculus. In McBurney, P., Rahwan, I., Parsons, S., and Maudet, N., editors, *Proceedings of the Sixth International Workshop on Argumentation in Multi-Agent Systems (ArgMAS 2009)*, pages 23–41.

Belesiotis, A., Rovatsos, M., and Rahwan, I. (2010). Agreeing on plans through iterated disputes. In van der Hoek, W., Kaminka, G. A., Lespérance, Y., Luck, M., and Sen, S., editors, *Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, pages 765–772.

Bertoli, P., Cimatti, A., Roveri, M., and Traverso, P. (2001). Planning in nondeterministic domains under partial observability via symbolic model checking. In Nebel, B., editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI 2001)*, pages 473–478.

Besnard, P. and Hunter, A. (2001). A logic-based theory of deductive arguments. *Artificial Intelligence*, 128(1-2):203–235.

Besnard, P. and Hunter, A. (2005). Practical first-order argumentation. In Veloso, M. M. and Kambhampati, S., editors, *Proceedings of the Twentieth National Conference on Artificial intelligence (AAAI 2005)*, pages 590–595. AAAI Press.

Besnard, P. and Hunter, A. (2008). *Elements of argumentation*, volume 47. MIT Press.

Black, E. and Hunter, A. (2007). A generative inquiry dialogue system. In Durfee, E. H., Yokoo, M., Huhns, M. N., and Shehory, O., editors, *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007)*, pages 241–248.

Black, E. and Hunter, A. (2009). An inquiry dialogue system. *Autonomous Agents and Multi-Agent Systems*, 19(2):173–209.

Blum, A. and Furst, M. (1995). Fast planning through planning graph analysis. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI 1995)*, pages 1636–1642.

Blum, A. and Furst, M. (1997). Fast planning through planning graph analysis. *Artificial Intelligence*, 90(1-2):281–300.

Bondarenko, A., Dung, P., Kowalski, R., and Toni, F. (1997). An abstract, argumentation-theoretic approach to default reasoning. *Artificial Intelligence*, 93(1-2):63–101.

Bonet, B. and Geffner, H. (2000). Planning with incomplete information as heuristic search in belief space. In Chien, S., Kambhampati, S., and Knoblock, C. A., editors,

*Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems (AIPS 2000)*, pages 52—61.

Bonet, B. and Geffner, H. (2001). Planning as heuristic search. *Artificial Intelligence*, 129(1-2):5–33.

Boutilier, C. and Brafman, R. (2001). Partial-order planning with concurrent interacting actions. *Journal of Artificial Intelligence Research*, 14(1):105–136.

Boutilier, C., Dean, T., and Hanks, S. (1999). Decision theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11(1):94.

Brafman, R. and Domshlak, C. (2008). From one to many: Planning for loosely coupled multi-agent systems. In Rintanen, J., Nebel, B., Beck, J. C., and Hansen, E. A., editors, *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling (ICAPS 2008)*, pages 28–35.

Brafman, R., Domshlak, C., Engel, Y., and Tennenholtz, M. (2009). Planning games. In Boutilier, C., editor, *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI 2009)*, pages 73–78.

Brafman, R. and Hoffmann, J. (2004). Conformant planning via heuristic forward search: A new approach. In Zilberstein, S., Koehler, J., and Koenig, S., editors, *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS 2004)*, pages 355—364.

Brewka, G. (2001). Original article dynamic argument systems: A formal model of argumentation processes based on situation calculus. *Journal of Logic and Computation*, 11(2):257–282.

Caminada, M. (2006). On the issue of reinstatement in argumentation. In Fisher, M., van der Hoek, W., Konev, B., and Lisitsa, A., editors, *Proceedings of the Tenth European Conference on Logics in Artificial Intelligence (JELIA 2006)*, pages 111–123.

Caminada, M. (2008). On the issue of contraposition of defeasible rules. In Besnard, P., Doutre, S., and Hunter, A., editors, *Proceedings of the Second International Conference on Computational Models of Argument (COMMA 2008)*, pages 109–115.

Clement, B., Barrett, A., and Schaffer, S. (2004). Argumentation for coordinating shared activities. In *Proceedings of the Fourth International Workshop on Planning and Scheduling for Space (IWPSS 2004)*.

Cohen, P. and Levesque, H. (1991a). Confirmations and joint action. In John Mylopoulos, R. R., editor, *Proceedings of the Twelfth International Joint Conference on Artificial intelligence (IJCAI 1991)*, pages 951–957.

Cohen, P. and Levesque, H. (1991b). Teamwork. *Noûs: Special Issue on Cognitive Science and Artificial Intelligence*, 25(4):487–512.

Conway, L., Lesser, V., and Corkill, D. (1983). The distributed vehicle monitoring testbed: A tool for investigating distributed problem solving networks. *AI Magazine*, 4(3):15.

Corkill, D. (1979). Hierarchical planning in a distributed environment. In *Proceedings of the Sixth International Joint Conference on Artificial Intelligence (IJCAI 1979)*, volume 1, pages 168–175.

Crosby, M. and Rovatsos, M. (2011). Heuristic multiagent planning with self-interested agents. In Sonenberg, L., Stone, P., Tumer, K., and Yolum, P., editors, *Proceedings of the Tenth International Joint Conference on Autonomous Agents Multiagent Systems (AAMAS 2011)*, pages 1213–1214.

Currie, K. and Tate, A. (1991). O-plan: the open planning architecture. *Artificial Intelligence*, 52(1):49–86.

de Weerdt, M. and Clement, B. (2009). Introduction to planning in multiagent systems. *Multiagent and Grid Systems*, 5(4):345–355.

Demolombe, R. and Parra, P. (2006). Belief revision in the situation calculus without plausibility levels. In Esposito, F., Ras, Z. W., Malerba, D., and Semeraro, G., editors, *Proceedings of the Sixteenth International Symposium on Foundations of Intelligent Systems (ISMIS 2006)*, pages 504–513.

DesJardins, M., Durfee, E., Ortiz Jr, C., and Wolverton, M. (1999). A survey of research in distributed, continual planning. *AI Magazine*, 20(4):13.

Dimopoulos, Y. and Kakas, A. (1995). Logic programming without negation as failure. In Lloyd, J. W., editor, *Proceedings of the 1995 International Symposium on Logic Programming (ILPS 1995)*, pages 369–384.

Drakengren, T. and Bjäreland, M. (1999). Reasoning about action in polynomial time. *Artificial Intelligence*, 115(1):1–24.

Dung, P., Kowalski, R., and Toni, F. (2009). Assumption-based argumentation. *Argumentation in Artificial Intelligence*, pages 199–218.

Dung, P. M. (1995). On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321–357.

Dunne, P. E. and Bench-Capon, T. (2003). Two party immediate response disputes: properties and efficiency. *Artificial Intelligence*, 149(2):221–250.

Durfee, E. (1999). Distributed Problem Solving and Planning. In Weiß, G., editor, *Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence*, chapter 3, pages 121–164. MIT Press.

Durfee, E. and Lesser, V. (1991). Partial global planning: A coordination framework for distributed hypothesis formation. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(5):1167–1183.

Eiter, T., Erdem, E., Fink, M., and Senko, J. (2010). Updating action domain descriptions. *Artificial Intelligence*, 174(15):1172–1221.

Ephrati, E., Pollack, M., and Rosenschein, J. (1995). A tractable heuristic that maximizes global utility through local plan combination. In Victor R. Lesser, L. G., editor, *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS 1995)*, pages 94–101.

Ephrati, E. and Rosenschein, J. (1993). Multi-agent planning as the process of merging distributed sub-plans. In Katia P. Sycara, M. F., editor, *In Proceedings of the Twelfth International Workshop on Distributed Artificial Intelligence (DAI 1993)*, pages 115—129.

Ephrati, E. and Rosenschein, J. (1994a). Divide and conquer in multi-agent planning. In Barbara Hayes-Roth, R. E. K., editor, *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI 1994)*, pages 375–375.

Ephrati, E. and Rosenschein, J. (1994b). Multi-agent planning as search for a consensus that maximizes social welfare. *Artificial Social Systems*, pages 207–226.

Erol, K., Hendler, J., and Nau, D. (1994). Umcp: A sound and complete procedure for hierarchical task-network planning. In Hammond, K. J., editor, *Proceedings of the Second International Conference on Artificial Intelligence Planning Systems (AIPS 1994)*, pages 249–254.

Falappa, M., Kern-Isberner, G., and Simari, G. (2009). Belief revision and argumentation theory. In Rahwan, I. and Simari, G., editors, *Argumentation in Artificial Intelligence*, pages 341–360. Springer.

Ferguson, G. and Allen, J. (1994). Arguing about plans: Plan representation and reasoning for mixed-initiative planning. In Hammond, K. J., editor, *Proceedings of the Second International Conference on AI Planning Systems (AIPS 1994)*, pages 43–48.

Fikes, R. and Nilsson, N. (1972). Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208.

Foulser, D., Li, M., and Yang, Q. (1992). Theory and algorithms for plan merging. *Artificial Intelligence*, 57(2-3):143–181.

Fox, J. and Das, S. (2000). *Safe and sound: artificial intelligence in hazardous applications*. AAAI Press.

Fox, J., Glasspool, D., Grecu, D., Modgil, S., South, M., and Patkar, V. (2007). Argumentation-based inference and decision making–a medical perspective. *IEEE Intelligent Systems*, 22(6):34–41.

García, A. and Simari, G. (2004). Defeasible logic programming: An argumentative approach. *Theory and Practice of Logic Programming*, 4(2):95–138.

García, D. R., García, A. J., and Simari, G. R. (2007). Planning and defeasible reasoning. In Durfee, E. H., Yokoo, M., Huhns, M. N., and Shehory, O., editors, *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007)*.

García, D. R., García, A. J., and Simari, G. R. (2008). Defeasible reasoning and partial order planning. In Hartmann, S. and Kern-Isberner, G., editors, *Proceedings of the Fifth International Symposium on Foundations of Information and Knowledge Systems (FoIKS 2008)*, pages 311–328.

Gelfond, M. and Lifschitz, V. (1990). Logic programs with classical negation. In Warren, D. H., editor, *Logic Programming*, pages 579–597. MIT Press.

Gelfond, M. and Lifschitz, V. (1993). Representing action and change by logic programs. *The Journal of Logic Programming*, 17(2-4):301–321.

Georgeff, M. P. (1983). Communication and Interaction in Multi-Agent Planning. In Genesereth, M. R., editor, *Proceedings of the Third National Conference on Artificial Intelligence (AAAI 1983)*, pages 125–129.

Giunchiglia, E. and Lifschitz, V. (1998). An action language based on causal explanation: Preliminary report. In Mostow, J. and Rich, C., editors, *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI 1998)*, pages 623–630.

Goldman, R. and Boddy, M. (1996). Expressive planning and explicit knowledge. In Drabble, B., editor, *Proceedings of the Third International Conference on Artificial Intelligence Planning Systems (AIPS 1996)*, pages 110–117.

Grosz, B. and Kraus, S. (1996). Collaborative plans for complex group action. *Artificial Intelligence*, 86(2):269–357.

Grosz, B. and Kraus, S. (1999). The evolution of sharedplans. In Wooldridge, M. J. and Rao, A., editors, *Foundations of Rational Agency*, pages 227–262. Kluwer.

Hamblin, C. (1970). *Fallacies*. Methuen.

Hoffmann, J. (2005). Where 'ignoring delete lists' works: Local search topology in planning benchmarks. *Journal of Artificial Intelligence Research*, 24(1):685–758.

Hoffmann, J. and Brafman, R. I. (2005). Contingent planning via heuristic forward search with implicit belief states. In Biundo, S., Myers, K. L., and Rajan, K., editors, *Proceedings of the Fifteenth International Conference on Automated Planning and Scheduling (ICAPS 2005)*, pages 71–80.

Hoffmann, J. and Nebel, B. (2001). The ff planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302.

Hulstijn, J. and van der Torre, L. (2004). Combining goal generation and planning in an argumentation framework. In Delgrande, J. P. and Schaub, T., editors, *Proceedings*

*of the Tenth International Workshop on Non-Monotonic Reasoning (NMR 2004)*, pages 212–218.

Jennings, N. (1995). Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence*, 75(2):195–240.

Jonsson, A. and Rovatsos, M. (2011). Scaling up multiagent planning: A best-response approach. In Bacchus, F., Domshlak, C., Edelkamp, S., and Helmert, M., editors, *Twenty-First International Conference on Automated Planning and Scheduling (ICAPS 2011)*, pages 114—121.

Kakas, A. and Miller, R. (1997a). Reasoning about actions, narratives and ramifications. *Electronic Transactions on Artificial Intelligence*, 1(4).

Kakas, A. and Miller, R. (1997b). A simple declarative language for describing narratives with actions. *The Journal of Logic Programming*, 31(1-3):157–200.

Kakas, A., Miller, R., and Toni, F. (1999). An argumentation framework for reasoning about actions and change. In Gelfond, M., Leone, N., and Pfeifer, G., editors, *Proceedings of the Fifth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 1999)*, pages 78–91.

Kakas, A., Miller, R., and Toni, F. (2001). E-res: Reasoning about actions, events and observations. In Eiter, T., Faber, W., and Truszczynski, M., editors, *Proceedings of the Sixth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2001)*, pages 254–266.

Katz, M. and Rosenschein, J. (1989). Plans for multiple agents. *Distributed Artificial Intelligence*, 2:197–228.

Kitano, H. and Tadokoro, S. (2001). Robocup rescue: A grand challenge for multiagent and intelligent systems. *AI Magazine*, 22(1):39.

Konolige, K. (1988). Defeasible argumentation in reasoning about events. In Raś, Z. and Zemankova, M., editors, *Proceedings of the Third International Symposium on Methodologies for Intelligent Systems (ISMIS 1998)*, pages 380–390.

Kowalski, R. and Sergot, M. (1986). A logic-based calculus of events. *New Generation Computing*, 4(1):67–95.

Kraus, S., Sycara, K., and Evenchik, A. (1998). Reaching agreements through argumentation: a logical model and implementation. *Artificial Intelligence*, 104(1-2):1–69.

Lansky, A. (1991). Localized search for multiagent planning. In John Mylopoulos, R. R., editor, *Proceedings of the Twelfth International Joint Conference on Artificial intelligence (IJCAI 1991)*, pages 252–258.

Larbi, R. B., Konieczny, S., and Marquis, P. (2007). Extending classical planning to the multi-agent case: A game-theoretic approach. In Mellouli, K., editor, *Proceedings of the Ninth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2007)*, pages 731–742.

Levesque, H. (1996). What is planning in the presence of sensing? In Clancey, W. J. and Weld, D. S., editors, *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI 1996)*, pages 1139–1146.

Levesque, H., Cohen, P., and Nunes, J. (1990). On acting together. In Shrobe, H. E., Dietterich, T. G., and Swartout, W. R., editors, *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI 1990)*, pages 94–99.

Lifschitz, V. (1996). Foundations of logic programs. *Principles of Knowledge Representation*, pages 69–128.

McAllester, D. and Rosenblitt, D. (1991). Systematic nonlinear planning. In Dean, T. L. and McKeown, K., editors, *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI 1991)*, volume 2, pages 634–639.

McBurney, P. and Parsons, S. (2002). Games that agents play: A formal framework for dialogues between autonomous agents. *Journal of Logic, Language and Information*, 11(3):315–334.

McBurney, P. and Parsons, S. (2009). Dialogue games for agent argumentation. In Rahwan, I. and Simari, G., editors, *Argumentation in Artificial Intelligence*, pages 261–280. Springer.

McCarthy, J. (1963). Situations, actions, and causal laws. Technical report, Stanford University. Reprinted in Semantic Information Processing (M. Minsky ed.), MIT Press, 1968, 410–417.

McCarthy, J. (1977). Epistemological problems of artificial intelligence. In Reddy, R., editor, *Proceedings of the Fifth International Joint Conference on Artificial Intelligence (IJCAI 1977)*, pages 1038–1044. Citeseer.

McCarthy, J. and Hayes, P. J. (1969). Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4:463–502.

McDermott, D. (1996). A heuristic estimator for means-ends analysis in planning. In Drabble, B., editor, *Proceedings of the Third International Conference on Artificial Intelligence Planning Systems (AIPS 1996)*, pages 142–149.

McDermott, D. (2000). The 1998 AI planning systems competition. *AI Magazine*, 21(2):35.

Medellin-Gasque, R., Atkinson, K., McBurney, P., and Bench-Capon, T. (2011). Arguments over co-operative plans. In Modgil, S., Oren, N., and Toni, F., editors, *Proceedings of the First International Workshop on Theory and Applications of Formal Argumentation (TAFA 2011)*, pages 50–66.

Modgil, S. and Caminada, M. (2009). Proof theories and algorithms for abstract argumentation frameworks. *Argumentation in AI*.

Morali, M. P. (2011). A demonstrator to "expose" agent-based argumentation. Master's thesis, School of Informatics, University of Edinburgh.

Moses, Y. and Tennenholtz, M. (1995). Multi-entity models. *Machine Intelligence*, 14:63–88.

Nau, D., Au, T., Ilghami, O., Kuter, U., Murdock, J., Wu, D., and Yaman, F. (2003). Shop2: An htn planning system. *Journal of Artificial Intelligence Research*, 20(1):379–404.

Nau, D., Ghallab, M., and Traverso, P. (2004). *Automated Planning: Theory & Practice*. Morgan Kaufmann.

Nebel, B. (2000). On the compilability and expressive power of propositional planning formalisms. *Journal of Artificial Intelligence Research*, 12:271–315.

Nissim, R., Brafman, R., and Domshlak, C. (2010). A general, fully distributed multi-agent planning algorithm. In van der Hoek, W., Kaminka, G. A., Lespérance, Y.,

Luck, M., and Sen, S., editors, *Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, pages 1323–1330.

Paglieri, F. and Castelfranchi, C. (2005). Revising beliefs through arguments: Bridging the gap between argumentation and belief revision in mas. In Rahwan, I., Moraitis, P., and Reed, C., editors, *Proceedings of the First International Workshop on Argumentation in Multi-Agent Systems (ArgMAS 2005)*, pages 78–94.

Palacios, H. and Geffner, H. (2009). Compiling uncertainty away in conformant planning problems with bounded width. *Journal of Artificial Intelligence Research*, 35(1):623–675.

Pardo, P., Pajares, S., Onaindia, E., Godo, L., and Dellunde, P. (2011). Multiagent argumentation for cooperative planning in delp-pop. In Sonenberg, L., Stone, P., Tumer, K., and Yolum, P., editors, *Proceedings of the Tenth International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, pages 971–978.

Parsons, S., Sierra, C., and Jennings, N. (1998). Agents that reason and negotiate by arguing. *Journal of Logic and Computation*, 8(3):261.

Pednault, E. (1989). ADL: Exploring the middle ground between STRIPS and the Situation Calculus. In Brachman, R. J., Levesque, H. J., and Reiter, R., editors, *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning (KR 1989)*, pages 324–332.

Petrick, R. and Bacchus, F. (2002). A knowledge-based approach to planning with incomplete information and sensing. In Ghallab, M., Hertzberg, J., and Traverso, P., editors, *Proceedings of the Sixth International Conference on Artificial Intelligence Planning Systems (AIPS 2002)*, pages 212–221.

Pollock, J. (1987). Defeasible reasoning. *Cognitive Science*, 11(4):481–518.

Pollock, J. (1998). The logical foundations of goal-regression planning in autonomous agents* 1. *Artificial Intelligence*, 106(2):267–334.

Pollock, J. (2001). Defeasible reasoning with variable degrees of justification. *Artificial Intelligence*, 133(1):233–282.

Poole, D. (1985). On the comparison of theories: Preferring the most specific explanation. In Joshi, A. K., editor, *Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI 1985)*, pages 144–147.

Popkorn, S. (1994). *First steps in modal logic*. Cambridge University Press.

Prakken, H. (2006). Formal systems for persuasion dialogue. *The Knowledge Engineering Review*, 21(2):163–188.

Prakken, H. and Sartor, G. (1997). Argument-based extended logic programming with defeasible priorities. *Journal of Applied Nonclassical Logics*, 7:25–76.

Prakken, H. and Vreeswijk, G. (2002). Logics for defeasible argumentation. In Gabbay, D. and Guenthner, F., editors, *Handbook of Philosophical Logic*, volume 4, pages 218–319. Kluwer.

Rahwan, I. and Amgoud, L. (2006). An argumentation based approach for practical reasoning. In Nakashima, H., Wellman, M. P., Weiss, G., and Stone, P., editors, *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006)*, pages 347–354.

Rahwan, I. and Simari, G. (2009). *Argumentation in artificial intelligence*. Springer.

Reiter, R. (1991). The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In Lifschitz, V., editor, *Artificial Intelligence and Mathematical Theory of Computation (Papers in Honor of John McCarthy)*, pages 359–380. Academic Press.

Reiter, R. (2001). *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press.

Röger, G., Helmert, M., and Nebel, B. (2008). On the relative expressiveness of ADL and Golog: The last piece in the puzzle. In Brewka, G. and Lang, J., editors, *Proceedings of the Eleventh International Conference on Principles of Knowledge Representation and Reasoning (KR 2008)*.

Rotstein, N. and Garcıa, A. (2006). Defeasible reasoning about beliefs and desires. In Dix, J. and Hunter, A., editors, *Proceedings of the Eleventh International Workshop on Non-Monotonic Reasoning (NMR 2011)*, pages 433–440.

Rotstein, N. D., García, A. J., and Simari, G. R. (2007). Reasoning from desires to intentions: a dialectical framework. In Collins, J., Faratin, P., Parsons, S., Rodríguez-Aguilar, J. A., Sadeh, N. M., Shehory, O., and Sklar, E., editors, *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence (AAAI 2007)*, pages 136–141.

Rotstein, N. D., García, A. J., and Simari, G. R. (2008). Defeasible argumentation support for an extended BDI architecture. In Rahwan, I., Parsons, S., and Reed, C., editors, *Proceedings of the Fourth International Workshop on Argumentation in Multi-Agent Systems (ArgMAS 2008)*, pages 145–163.

Russell, S. and Norvig, P. (2003). *Artificial Intelligence: a Modern Approach*. Prentice Hall, 2nd edition edition.

Sacerdoti, E. (1975). The nonlinear nature of plans. In *Proceedings of the Fourth International Joint Conference on Artificial Intelligence (IJCAI 1975)*, pages 206–214.

Sandewall, E. (1994). *Features and Fluents: The representation of knowledge about dynamical systems*. Clarendon Press.

Scherl, R. and Levesque, H. (2003). Knowledge, action, and the frame problem. *Artificial Intelligence*, 144(1-2):1–39.

Shapiro, S., Pagnucco, M., Lespérance, Y., and Levesque, H. (2011). Iterated belief change in the situation calculus. *Artificial Intelligence*, 175(1):165–192.

Simari, G., Garcia, A., and Capobianco, M. (2004). Actions, planning and defeasible reasoning. In Delgrande, J. P. and Schaub, T., editors, *Proceedings of the Tenth International Workshop on Non-Monotonic Reasoning (NMR 2004)*, pages 377–384.

Simari, G. and Loui, R. (1992). A mathematical treatment of defeasible reasoning and its implementation. *Artificial Intelligence*, 53(2-3):125–157.

Smith, D. and Weld, D. (1998). Conformant graphplan. In Mostow, J. and Rich, C., editors, *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI 1998)*, pages 889–896.

Stuart, C. (1985). An implementation of a multi-agent plan synchronizer. In Joshi, A. K., editor, *Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI 1985)*, pages 1031–1035.

Sycara, K. (1989). Argumentation: Planning other agents' plans. In Sridharan, N. S., editor, *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI 1989)*, pages 517–523.

Tambe, M. (1997). Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7:83–124.

Tambe, M. and Jung, H. (1999). The benefits of arguing in a team. *AI Magazine*, 20(4):85.

Tang, Y. (2012). *A Symbolic Exploration of the Joint State Space and the Underlying Argumentation-based Reasoning Processes for Multiagent Planning*. PhD thesis, The City University of New York.

Tang, Y., Norman, T. J., and Parsons, S. (2009). A model for integrating dialogue and the execution of joint plans. In Decker, K. S., Sichman, J. S., Sierra, C., and Castelfranchi, C., editor, *Proceedings of the Eigth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, pages 883–890.

Tang, Y. and Parsons, S. (2005). Argumentation-based dialogues for deliberation. In Dignum, F., Dignum, V., Koenig, S., Kraus, S., Singh, M. P., and Wooldridge, M., editors, *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005)*, pages 552–559.

Tate, A. (1977). Generating project networks. In Reddy, R., editor, *Proceedings of the Fifth International Joint Conference on Artificial Intelligence (IJCAI 1977)*, pages 888–893.

Toniolo, A., Norman, T. J., and Sycara, K. (2011). Argumentation schemes for collaborative planning. In Kinny, D., jen Hsu, J. Y., Governatori, G., and Ghose, A. K., editors, *Proceedings of the Fourteenth International Conference on Principles and Practice of Multi-Agent Systems (PRIMA 2011)*, pages 323–335.

Toulmin, S. (1958). *The Uses of Argument*. Cambridge University Press.

Tsamardinos, I., Pollack, M., and Horty, J. (2000). Merging plans with quantitative temporal constraints, temporally extended actions, and conditional branches. In Chien, S., Kambhampati, S., and Knoblock, C. A., editors, *Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems (AIPS 2000)*, pages 264–272.

Van Der Hoek, W. and Wooldridge, M. (2002). Tractable multiagent planning for epistemic goals. In Castelfranchi, C., Gini, M., Ishida, T., and Johnson, W. L., editors,

*Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2002)*, pages 1167–1174.

Varzinczak, I. (2010). On action theory change. *Journal of Artificial Intelligence Research*, 37:189–246.

Vo, Q. and Foo, N. (2001). Solving the qualification problem. In Stumptner, M., Corbett, D., and Brooks, M. J., editors, *Proceedings of the Fourteenth Australian Joint Conference on Artificial Intelligence (AI 2001)*, pages 519–531.

Vo, Q. and Foo, N. (2002). Solving the ramification problem: Causal propagation in an argumentation-theoretic approach. In Ishizuka, M. and Sattar, A., editors, *Proceedings of the Seventh Pacific Rim International Conference on Artificial Intelligence (PRICAI 2002)*, pages 49–59.

Vo, Q. and Foo, N. (2005). Reasoning about action: An argumentation-theoretic approach. *Journal of Artificial Intelligence Research*, 24:465–518.

Vreeswijk, G. and Prakken, H. (2000). Credulous and sceptical argument games for preferred semantics. In Ojeda-Aciego, M., de Guzmán, I. P., Brewka, G., and Pereira, L. M., editors, *Proceedings of the Sixth European Conference on Logics in Artificial Intelligence (JELIA 2000)*, pages 239–253.

Walton, D. and Krabbe, E. (1995). *Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning*. State University of New York Press.

Walton, D. N. (1996). *Argumentation Schemes for Presumptive Reasoning*. Lawrence Erlbaum Associates.

Weiss, G. (1999). *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press.

Weld, D. (1999). Recent advances in AI planning. *AI Magazine*, 20(2):93–123.

Werner, E. (1988). Toward a theory of communication and cooperation for multiagent planning. In Vardi, M. Y., editor, *Proceedings of the Second Conference on Theoretical Aspects of Reasoning about Knowledge (TARK 1988)*, pages 129–143.

Wolverton, M. and DesJardins, M. (1998). Controlling communication in distributed planning using irrelevance reasoning. In Mostow, J. and Rich, C., editors, *Proceed-*

*ings of the Fifteenth National Conference on Artificial Intelligence (AAAI 1998)*, pages 868–874.

Wooldridge, M. (2002). *An Introduction to MultiAgent Systems*. John Wiley and Sons.

Wooldridge, M. and Jennings, N. (1999). The cooperative problem-solving process. *Journal of Logic and Computation*, 9(4):563.

Yang, Q., Nau, D., and Hendler, J. (1992). Merging separately generated plans with restricted interactions. *Computational Intelligence*, 8(4):648–676.

Zhang, Y. (2003). Handling defeasibilities in action domains. *Theory and practice of logic programming*, 3(3):329–376.